PROGRAMLAMA DİLLERİ MODÜLÜ (PHP)

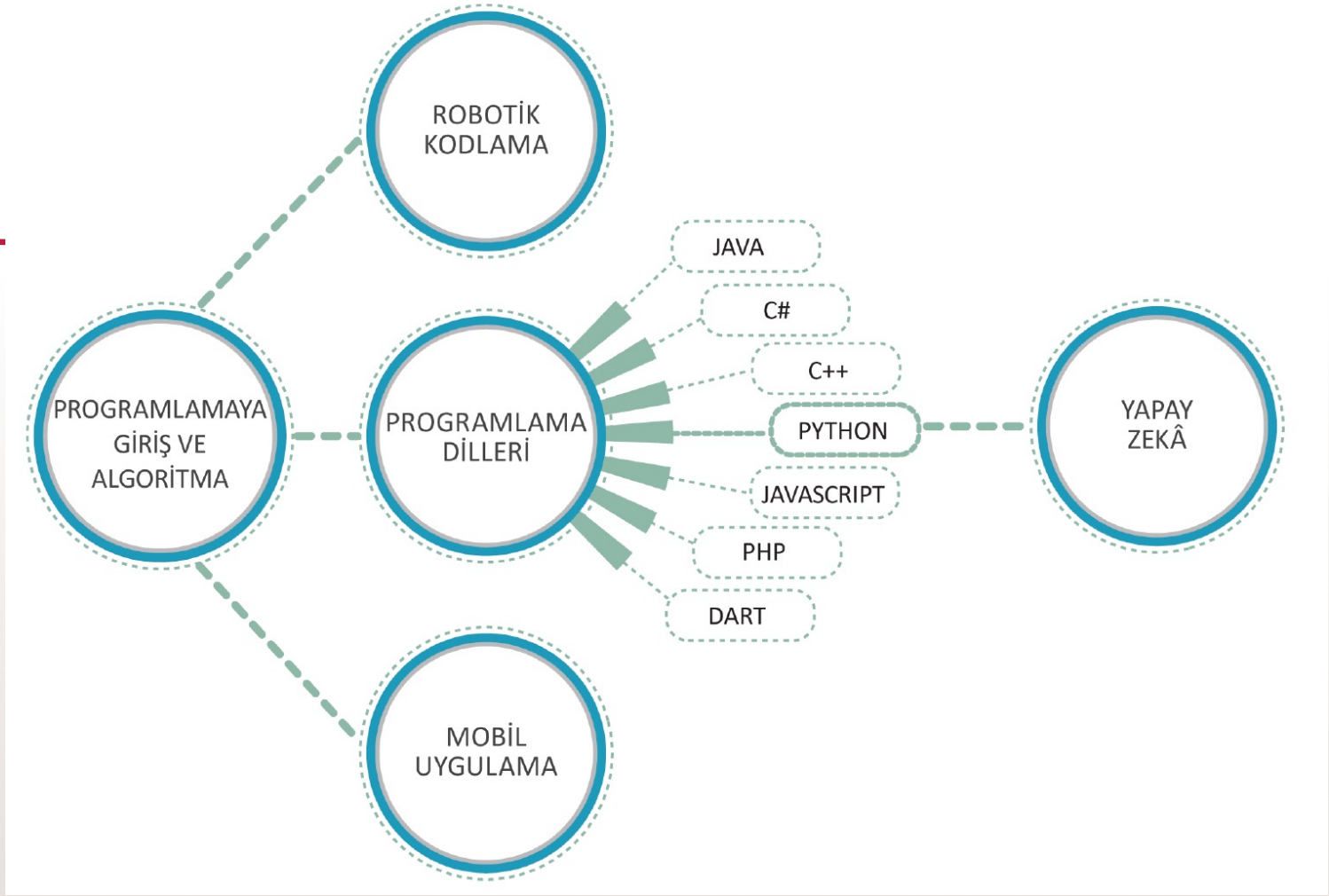
HAZ: ONUR ALTUNTAŞ

DERS SEÇİM KRİTERİ

Programlama dilleri modülünde; Java, C++, C#, Dart, PHP, Javascript, Python programlama dillerinden birisi seçilir.

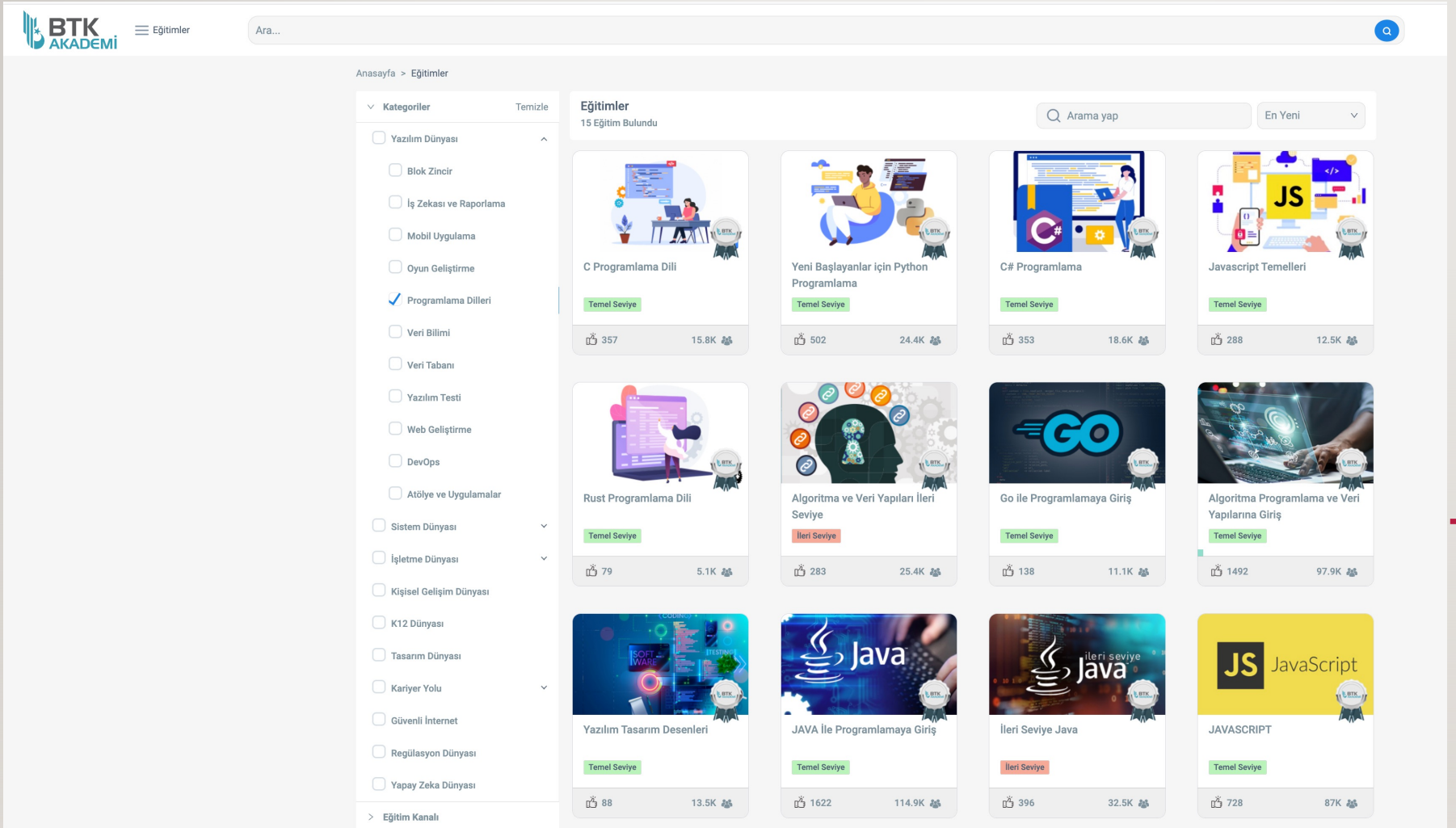
Programlamaya giriş ve algoritma modülü alınmadan; robotik kodlama, mobil uygulama geliştirme ve programlama dilleri modülleri seçilemez. Programlama dilleri modülü alınmadan yapay zekâ uygulamaları modülü seçilemez.

DERS SEÇİM KRİTERİ ŞEMASI

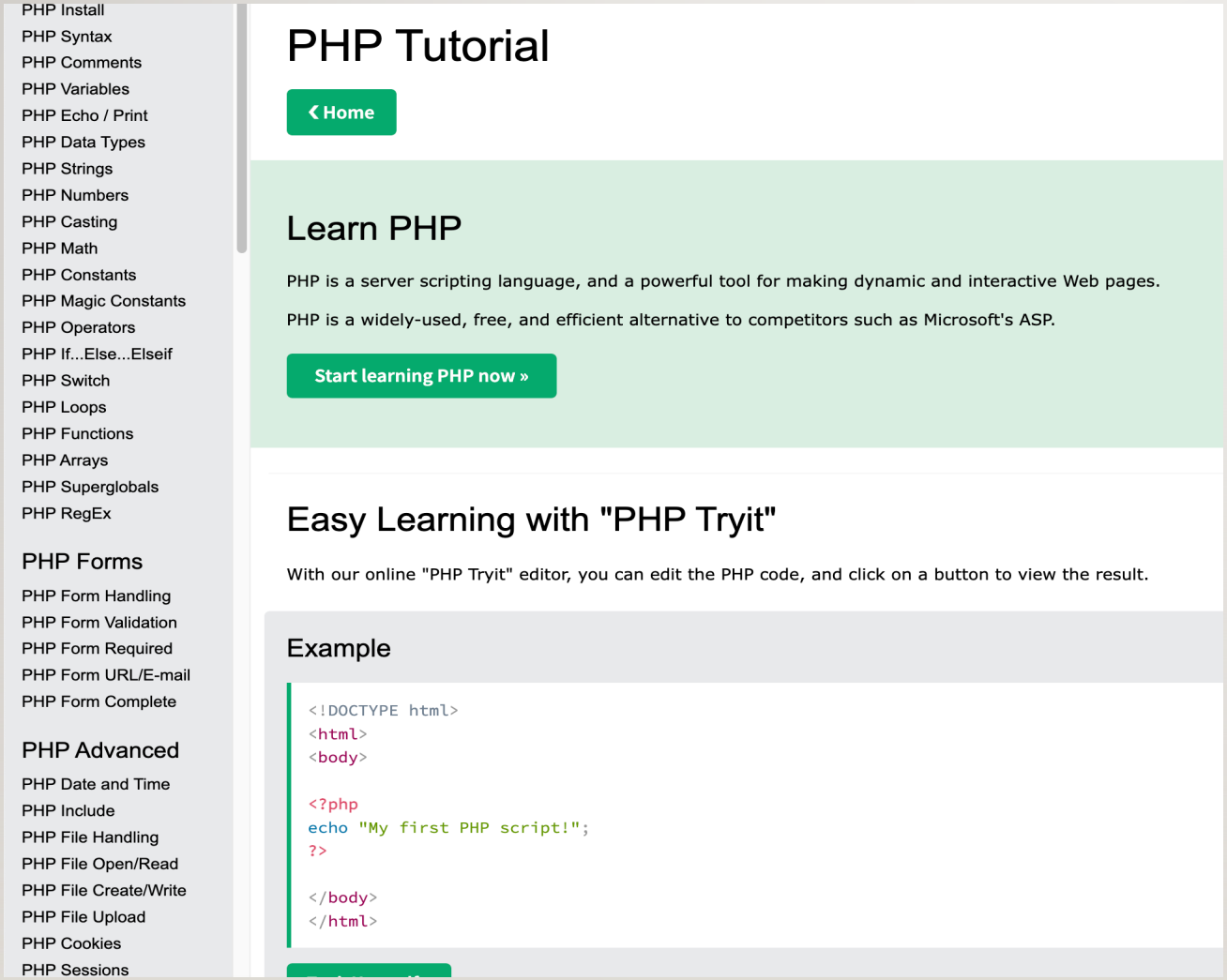


MODÜL	ÜNİTE	KAZA-NIM SAYISI	DERS SAATİ	YÜZDE ORANI (%)
PROGRAMLAMA DİLLERİ (PHP)	1. Ünite: PHP Temel Kavramlar	4	18	25
	2. Ünite: PHP'yi Tanıyalım	3	18	25
	3. Ünite: PHP Fonksiyonları, Aritmetiksel ve Mantıksal işlemler	3	18	25
	4. Ünite: PHP ile Veri Tabanı İşlemleri	3	18	25

ÖĞRENME KAYNAKLARI



w3schools.com



ÖĞRENME KAYNAKLARI

PHP

PHP (Hypertext Preprocessor), dinamik ve etkileşimli web sayfaları oluşturmak için kullanılan popüler bir genel amaçlı betik (script) dilidir.

PHP MODÜLÜNÜN AMACI

- PHP programlama dilinde temel kavramlar, çalışma mantığı, yazım kuralları, fonksiyonlar ve veri tabanı işlemleri gibi konuların ayrıntılı biçimde ele alınması amaçlanmaktadır.

PHP

ÜNİTE - KAZANIM VE AÇIKLAMALARI

1. ÜNİTE: PHP TEMEL KAVRAMLAR

Ünite Açıklaması

- Bu ünitede, PHP programlama dilinin hangi ortamda çalıştığı ve nasıl kurulacağı konularına değinilmiştir.

Değerler

- Sabır (Kazanım 1.1., 1.3.)
- Öz Denetim (Kazanım 1.3.)

Alan Becerileri

- Soyutlama Becerisi (Kazanım 1.1.)
- Teknik İşlem Becerisi (Kazanım 1.3.)

1.1. SUNUCU VE İSTEMCİ TARAFINDA ÇALIŞMA MANTIĞINI KAVRAR.

a) Sunucu ve istemci kavramları açıklanır.

b) Sunucu tarafında çalışan yapılardan bahsedilir. Javascript kısaca açıklanır ve PHP programlama dili ile arasındaki farklar anlatılır.

c) Sunucu ve istemci tarafında çalışan yapıların farkları anlatılır.

1.1. A) SUNUCU VE İSTEMCİ KAVRAMLARI AÇIKLANIR.

Sunucu, ağ üzerinden istemcilere veri veya hizmet sağlayan bir bilgisayar veya yazılım sistemidir. Sunucular, dosyaları, web sayfalarını, uygulamaları ve diğer kaynakları barındırabilir ve bu kaynaklara erişim isteklerine yanıt verir. Örneğin, bir web sunucusu, internet üzerinden web sayfalarını isteyen tarayıcılara bu sayfaları sunar. Sunucular genellikle güçlü işlem kapasitesine, büyük miktarda belleğe ve geniş depolama alanına sahiptir, böylece birden fazla istemciye eş zamanlı olarak hizmet verebilirler.

1.1. A) SUNUCU VE İSTEMCİ KAVRAMLARI AÇIKLANIR.

İstemci, kullanıcının bir sunucudan veri veya hizmet talep etmek için kullandığı bir bilgisayar veya yazılım sistemidir. Bu terim, genellikle bu istekleri yapmak için kullanılan yazılımları ifade eder; örneğin, bir web tarayıcısı bir web sunucusundan web sayfaları istemek için kullanılan bir istemcidir. İstemciler, genellikle son kullanıcılar tarafından kullanılan masaüstü bilgisayarlar, dizüstü bilgisayarlar, akıllı telefonlar gibi cihazlardır.

1.1. B) SUNUCU TARAFINDA ÇALIŞAN YAPILARDAN BAHSEDİLİR. JAVASCRİPT KISACA AÇIKLANIR VE PHP PROGRAMLAMA DİLİ İLE ARASINDAKİ FARKLAR ANLATILIR.

- PHP, Python, Ruby, Java, Node.js gibi teknolojiler sunucu tarafında yaygın olarak kullanılan araçlardır. Bu teknolojiler, web sayfalarının dinamik olarak oluşturulmasını, kullanıcı bilgilerinin işlenmesini, veritabanı yönetimini ve API (Uygulama Programlama Arayüzü) hizmetlerinin sağlanmasını mümkün kılar.
- JavaScript, başlangıçta yalnızca istemci tarafında (tarayıcıda) çalışacak şekilde tasarlanmış bir betik dilidir. Web sayfalarını interaktif hale getirmek için kullanılır; kullanıcı etkileşimlerine yanıt vermek, dinamik içerik üretmek, animasyonlar eklemek gibi görevleri yerine getirir. Son yıllarda, Node.js'in geliştirilmesiyle birlikte JavaScript, sunucu tarafında da kullanılmaya başlanmış ve böylece JavaScript ile tam yığın (full-stack) geliştirme mümkün hale gelmiştir.

1.1. B) SUNUCU TARAFINDA ÇALIŞAN YAPILARDAN BAHSEDİLİR. JAVASCRİPT KISACA AÇIKLANIR VE PHP PROGRAMLAMA DİLİ İLE ARASINDAKİ FARKLAR ANLATILIR.

- PHP, bir restoranın mutfağı gibidir. Sipariş verdiğinizde (web sitesine bir istek gönderdiğinizde), mutfak (sunucu) siparişinizi hazırlar ve size hazır bir yemek (web sayfası) sunar. Yani, PHP kodları web sunucusunda çalışır; siz onları doğrudan göremezsiniz, sadece sonuçlarını (örneğin, bir web sayfası) görürsünüz. PHP, genellikle veritabanlarıyla çalışır ve dinamik içerik oluşturur - örneğin, bir kullanıcı adıyla kişisel bir karşılama mesajı gösterir.
- JavaScript ise restoranda sizin masanızda olan bitendir. Yemeğinizi aldıktan sonra, ekstra tuz eklemek veya karabiber dökmek gibi işlemler yapabilirsiniz. JavaScript, kullanıcının tarayıcısında çalışır ve sayfa yüklendikten sonra sayfayı dinamik bir şekilde değiştirebilir. Örneğin, bir butona tıklandığında bir menünün açılması gibi etkileşimleri sağlar.

1.1. B) SUNUCU TARAFINDA ÇALIŞAN YAPILARDAN BAHSEDİLİR. JAVASCRİPT KISACA AÇIKLANIR VE PHP PROGRAMLAMA DİLİ İLE ARASINDAKİ FARKLAR ANLATILIR.

Basitçe Farklar

- **Çalışma Yeri:** PHP sunucuda çalışırken, JavaScript kullanıcının bilgisayarında (tarayıcıda) çalışır.
- **Görevler:** PHP genellikle sayfa oluşturma ve veritabanı işlemleri için kullanılır. JavaScript ise sayfa üzerindeki etkileşimleri (örneğin, tıklamalar, animasyonlar) yönetir.
- **Görünürlük:** PHP'nin yaptığı işlemler kullanıcı tarafından doğrudan görülemez; sonuçları görülür. JavaScript ile yapılan değişiklikler ise kullanıcı tarafından anında görülebilir.
- Örnekle açıklamak gerekirse, bir sosyal medya sitesinde profilinize giriş yaptığınızda, PHP sizin kullanıcı adınızı veritabanından alıp sayfaya yerleştirir. Daha sonra, bir arkadaşınızın fotoğrafını beğenmek için kalp ikonuna tıklarsanız, bu etkileşim JavaScript ile sağlanır ve sayfa yeniden yüklenmeden kalp ikonunun rengi değişir.
- Böylece, PHP ve JavaScript birlikte çalışarak web sitelerinin hem arka planında (mutfakta) hem de ön yüzünde (masada) sorunsuz bir deneyim sunar.

1.1. C) SUNUCU VE İSTEMCİ TARAFINDA ÇALIŞAN YAPILARIN FARKLARI ANLATILIR.

Sunucu (Postane)

- Sunucu, bir postane gibidir. Postane, insanların mektup ve paket gönderip alabileceği bir yerdir. Burada, gelen ve giden her türlü posta işlenir, sıralanır ve doğru adreslere yönlendirilir. Sunucu da aynı şekilde, internet üzerinden gelen istekleri alır (web sayfası istekleri, e-postalar vs.), bunları işler ve gereken yanıtları geri gönderir. Sunucu, bu veri ve içeriği saklayan ve yöneten bir "merkez"dir.

İstemci (Müşteri)

- İstemci, postaneye gelen bir müşteri gibidir. Bir müşteri gibi, istemci (bilgisayarınız, telefonunuz veya herhangi bir internete bağlı cihaz) sunucuya (postaneye) bir istek gönderir. Bu, bir web sayfası açmak, bir video izlemek veya bir e-posta göndermek olabilir. Ardından, sunucunun (postanenin) verdiği yanıtı alır. İstemci, bu yanıtı kullanarak kullanıcıya bilgi gösterir veya bir hizmet sağlar.

1.1. C) SUNUCU VE İSTEMCİ TARAFINDA ÇALIŞAN YAPILARIN FARKLARI ANLATILIR.

Ana Farklar

- **Görev ve İşlev:** Sunucu, veri saklar, işler ve yönetir; isteklere yanıt verir. İstemci ise, kullanıcı adına istek gönderir ve sunucudan aldığı yanıtları kullanıcıya sunar.
- **Konum ve Erişim:** Sunucu genellikle uzak bir konumda bulunur ve birçok istemciye aynı anda hizmet verebilir. İstemci, kullanıcının doğrudan erişimine açık olan ve sunucuya istek gönderen cihazdır.
- **Kaynak Kullanımı:** Sunucular, genellikle yüksek işlem gücü, bellek ve depolama kapasitesine sahip olacak şekilde tasarlanmıştır çünkü birçok istemciye hizmet vermek zorundadırlar. İstemciler ise, daha az kaynak gerektiren işlemler için kullanılır ve genellikle tek bir kullanıcı tarafından kullanılır.

1.2. PHP PROGRAMLAMA DİLİ İÇİN GEREKLİ PROGRAMLARI TANIR.

a) PHP programlama dilinin çalışması için gerekli olan programlar tanıtılır.

b) Gerekli olan programların işlevleri açıklanır.

c) Her bir programın ne işlem yaptığından bahsedilir.

1.2. PHP PROGRAMLAMA DİLİ İÇİN GEREKLİ PROGRAMLARI TANIR.

1. **PHP (Şef) -** PHP.net
2. **Web Sunucusu (Mutfak)** Web sunucusu, PHP kodlarının çalıştırıldığı mutfaktır. En popüler web sunucuları Apache ve Nginx'dir. https://httpd.apache.org/download.cgi
3. **Veritabanı (Depo)** PHP ile en yaygın olarak kullanılan veritabanı MySQL'dir, ancak MariaDB, PostgreSQL gibi diğer veritabanları da kullanılabilir. https://www.mysql.com/downloads/
4. **PHP Geliştirme Ortamı – Editör (Aşçı Tezgahı)** PHP kodlarını yazmak, test etmek ve hata ayıklamak için bir geliştirme ortamına ihtiyacınız vardır. Basit bir metin editörü kullanabilirsiniz, ancak daha gelişmiş özellikler sunan IDE'ler (Entegre Geliştirme Ortamları) gibi araçlar işinizi kolaylaştırır. Örnek: PhpStorm, Visual Studio Code, Atom

1.2. PHP PROGRAMLAMA DİLİ İÇİN GEREKLİ PROGRAMLARI TANIR.

XAMPP/WAMP/LAMP/MAMP (Hazır Mutfak)

- Eğer tüm bu bileşenleri tek tek kurmak istemiyorsanız, XAMPP (Windows, Linux, macOS için), WAMP (sadece Windows için), LAMP (Linux için) veya MAMP (macOS için) gibi "tümü bir arada" çözümleri tercih edebilirsiniz. Bu paketler, PHP, bir web sunucusu ve MySQL veritabanını içerir ve kolay kurulumlarıyla bilinirler. Bu paketler, PHP tabanlı uygulamaları geliştirmek için ihtiyacınız olan "mutfak"ı hızlıca kurmanıza yardımcı olur.

1.3. PHP PROGRAMLAMA DİLİ İÇİN GEREKLİ PROGRAMLARIN KURULUMUNU YAPAR.

a) Bilgisayarda uygulamalı olarak gerekli programların kurulumu yaptırılır.

b) Kurulum adımları açıklanarak, farklı yol ve yöntemlerin kullanılabileceğinden bahsedilir

1.4. PHP PROGRAMLAMA DİLİNİN FARKLI İŞLETİM SİSTEMLERİNDE NASIL ÇALIŞTIĞINI KAVRAR.

a) PHP programlama dilinin çalışabildiği her bir işletim sisteminde nasıl çalıştığından bahsedilir.

b) İşletim sistemlerinde PHP programlama dilinin çalışmasındaki farklar, avantajlar ve dezavantajlar anlatılır.

1.4. PHP PROGRAMLAMA DİLİNİN FARKLI İŞLETİM SİSTEMLERİNDE NASIL ÇALIŞTIĞINI KAVRAR.

Windows İçin

- Windows'ta PHP çalıştırmak istiyorsan, bir partiye ev sahipliği yapmak gibi düşünebilirsin. Parti için bir ev (sunucu yazılımı gibi **WAMP**/ **XAMPP** : Windows, Apache, MySQL, PHP) ve partiye davet etmek istediğin insanlar (PHP kodları) gerekiyor. Bunlar bu parti için her şeyi bir arada sunan bir pakettir.
- Kurulum bittiğinde Windows bilgisayarın bir web sunucusuna dönüşür ve PHP kodlarını çalıştırabilirsin. Bu, evindeki partiye gelen misafirlerin eğlenmesi için gerekli her şeyi sağlamak gibidir.

1.4. PHP PROGRAMLAMA DİLİNİN FARKLI İŞLETİM SİSTEMLERİNDE NASIL ÇALIŞTIĞINI KAVRAR.

macOS İçin

- macOS'te PHP çalıştırmak, bir sanat galerisi açmaya benzer. Galerinde sergilemek istediğin eserler (PHP kodları) var. MAMP (Mac, Apache, MySQL, PHP) kullanarak, bu sanat galerisini açabilirsin. MAMP, macOS için geliştirilmiş, PHP dahil her şeyi içeren bir pakettir. MAMP'ı kurduğunda, Mac'in bir web sunucusuna dönüşür ve PHP kodlarını çalıştırabilir, böylece sanatını (web siteni) herkese açık bir şekilde sergileyebilirsin.

1.4. PHP PROGRAMLAMA DİLİNİN FARKLI İŞLETİM SİSTEMLERİNDE NASIL ÇALIŞTIĞINI KAVRAR.

Linux İçin

- Linux'ta PHP çalıştırmak, kendi kamp alanını kurmaya benzetilebilir. Kamp için çadır (Apache veya Nginx gibi bir web sunucusu), uyku tulumu (PHP) ve yemek malzemeleri (MySQL veya MariaDB gibi bir veritabanı sistemi) gerekiyor. Linux, genellikle bu araçları ayrı ayrı sağlar ve senin bunları bir araya getirip kurman gerekir. Bu, kamp alanını kendin kurmak gibi, biraz daha fazla çaba gerektirse de, sonunda tam istediğin gibi bir çalışma ortamın olur.

2. ÜNİTE: PHP PROGRAMLAMA DİLİNİ TANIYALIM

Ünite Açıklaması

- Bu ünitede, PHP programlama dilinin çalışma mantığı, yazım kuralları ve yapılandırma ayarları konularına değinilmiştir.

Değerler

- Sabır (Kazanım 2.1., 2.2.)
- Sorumluluk (Kazanım 2.3.)

Alan Becerileri

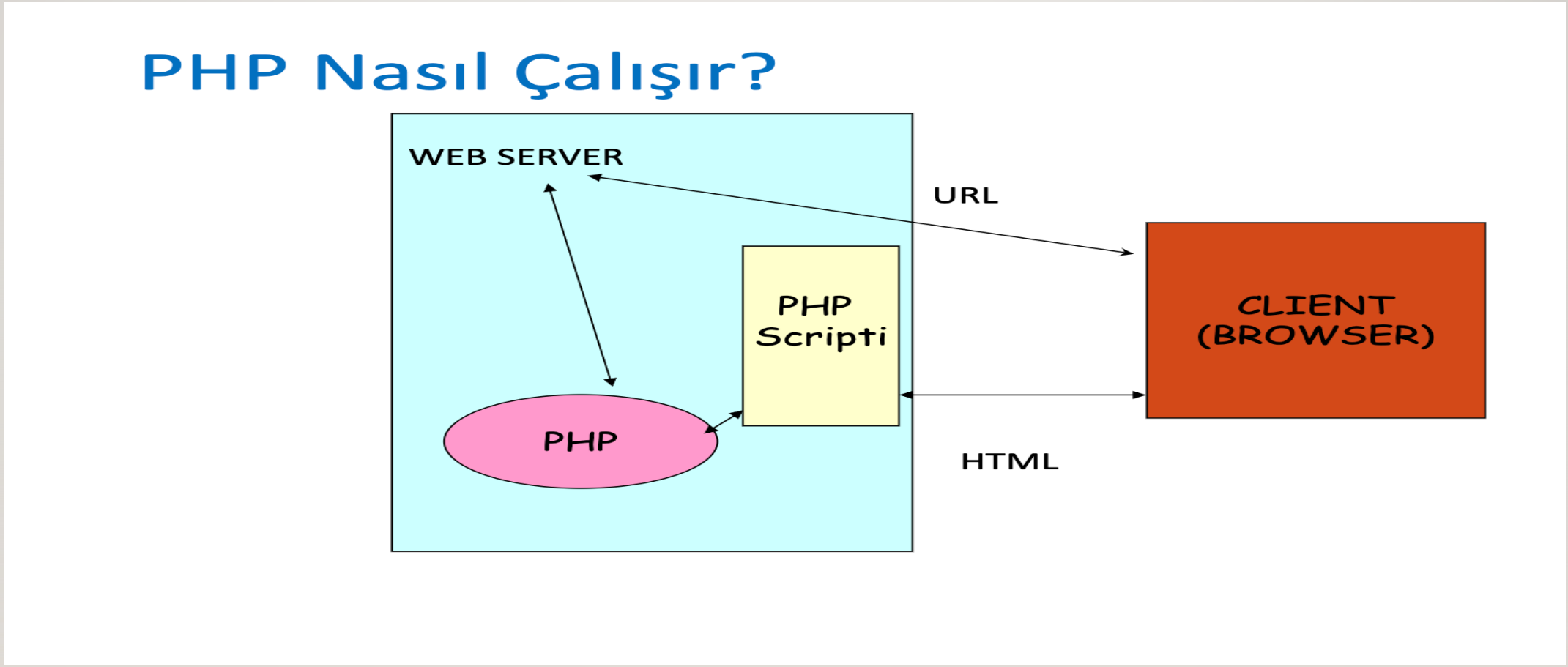
- Gözlem Yapma Becerisi (Kazanım 2.1.)
- Teknik İşlem Becerisi (Kazanım 2.3.)

2.1. PHP PROGRAMLAMA DİLİNİN NASIL ÇALIŞTIĞINI KAVRAR

a) PHP programlama dilinin nasıl çalıştığı görsel olarak anlatılır.

b) Kullanıcının yapmak istediği iş ve işlemler hangi yollardan geçerek sonuca ulaştığı anlatılır.

2.1. A) PHP PROGRAMLAMA DİLİNİN NASIL ÇALIŞTIĞI GÖRSEL OLARAK ANLATILIR.



2.1. B) KULLANICININ YAPMAK İSTEDİĞİ İŞ VE İŞLEMLER HANGİ YOLLARDAN GEÇEREK SONUCA ULAŞTIĞI ANLATILIR.

- Sunucu kurulduktan sonra açılıp çalıştırılır ve localhost portu ayarlanır.
- Sunucu kurulduktan sonra localhostun çalıştığı htdocs klasöründe bir çalışma klasörü oluşturulur.
- Çalışma klasöründe index.php adlı ana dosya oluşturulur.

2.1. B) KULLANICININ YAPMAK İSTEDİĞİ İŞ VE İŞLEMLER HANGİ YOLLARDAN GEÇEREK SONUCA ULAŞTIĞI ANLATILIR.

1. Sipariş Verme (Kullanıcı İsteği Gönderme)

- Bu adım, pizza siparişi vermek istediğinizde telefonu elinize alıp pizzacıyı aramanız gibidir. Web dünyasında bu, bir web formu doldurup 'Gönder' butonuna tıklamak veya bir web sayfasını ziyaret etmekle eşdeğerdir. Kullanıcı, bir web tarayıcısı (Chrome, Firefox vs.) aracılığıyla PHP kodlarının çalıştığı bir web sunucusuna bir istek gönderir.

2.1. B) KULLANICININ YAPMAK İSTEDİĞİ İŞ VE İŞLEMLER HANGİ YOLLARDAN GEÇEREK SONUCA ULAŞTIĞI ANLATILIR.

2. Siparişi Hazırlama (Sunucu İşleme)

- Pizzacıya siparişinizi verdikten sonra, pizzacı siparişinizi alır ve pizzayı hazırlamaya başlar. Web dünyasında, sunucu kullanıcının isteğini alır ve PHP kodlarını çalıştırır. Bu, bir veritabanından bilgi çekmek, kullanıcı verilerini işlemek veya belirli bir işlemi gerçekleştirmek olabilir. PHP kodları, istenilen işlemleri yaparak gerekli verileri hazırlar ve bir sonuç üretir.

2.1. B) KULLANICININ YAPMAK İSTEDİĞİ İŞ VE İŞLEMLER HANGİ YOLLARDAN GEÇEREK SONUCA ULAŞTIĞI ANLATILIR.

3. Siparişin Teslimi (Sonucun Kullanıcıya Gönderilmesi)

- Pizza hazırlandığında, pizzacı onu bir kutuya koyar ve adresinize gönderir. Benzer şekilde, PHP kodlarının işlenmesi tamamlandığında, sunucu sonucu (genellikle HTML formunda) kullanıcının tarayıcısına geri gönderir. Kullanıcı, tarayıcısında bu sonucu görür - bu, web sayfası formunda olabilir, bir onay mesajı veya kullanıcının isteğine bağlı olarak değişiklik gösteren herhangi bir içerik olabilir.

2.1. B) KULLANICININ YAPMAK İSTEDİĞİ İŞ VE İŞLEMLER HANGİ YOLLARDAN GEÇEREK SONUCA ULAŞTIĞI ANLATILIR.

4. Siparişin Yenilmesi (Sonucun Kullanıcı Tarafından Görüntülenmesi)

- Evinize ulaşan pizza kutusunu açıp pizzanızı yemeniz gibi, web sayfasının sonucunu da tarayıcınızda görüntüleyip bilgileri okuyabilir veya sayfadaki etkileşimlere katılabilirsiniz. Bu adım, PHP tarafından hazırlanan sonucun kullanıcı tarafından görüntülenmesi ve kullanılmasıdır.
- Bu süreçte, her adım bir sonrakini tetikler ve kullanıcının başlangıçtaki isteği, PHP kodlarının işlenmesiyle sonuçlanır. Kullanıcı, bu sonucu web tarayıcısında görür ve böylece sipariş verme süreci tamamlanmış olur. Bu, PHP'nin web geliştirme dünyasında nasıl önemli bir rol oynadığını gösterir: Kullanıcıların isteklerini almak, bu isteklere göre işlem yapmak ve sonuçları kullanıcılara sunmak.

2.2. YAZIM KURALLARINI KAVRAR.

a) $ karakterinin neden kullanıldığı anlatılır.

b) Kod yazımında parantezlerin ve noktalama işaretlerinin nerelerde ve nasıl kullanılacağı açıklanır.

c) Değişken tanımlama kuralları açıklanır.

ç) Veri türleri açıklanır.

2.2. A) $ KARAKTERİNİN NEDEN KULLANILDIĞI ANLATILIR.

- PHP'de **$** karakteri, bir değişkenin başında kullanılır ve bu değişkenin bir değeri sakladığını belirtir. Değişken, programınızda kullanabileceğiniz ve değişebilen bir veri parçasıdır. Örneğin, bir kişinin adını, bir sayıyı veya daha karmaşık veri yapılarını saklamak için değişkenleri kullanabilirsiniz.
- **$** işareti bir kutunun üzerine isim yazmak gibidir. Diyelim ki bir kutunun içine kitaplarınızı koydunuz ve bu kutuya "Kitaplar" diye bir etiket yapıştırdınız. Programlama dili PHP'de de, bir veri parçasını bir kutuya koymak ve bu kutuya bir isim vermek istediğinizde, bu ismin başına **$** işareti koyarsınız. Böylece, programınızın başka bir yerinde bu kutuya ihtiyacınız olduğunda, kutunun üzerindeki ismi (**$** işaretiyle birlikte) kullanarak o kutuya ulaşabilir ve içindeki verileri kullanabilirsiniz.

2.2. A) $ KARAKTERİNİN NEDEN KULLANILDIĞI ANLATILIR.

- Örneğin, bir kullanıcının adını saklamak istiyorsanız, şöyle bir PHP kodu yazabilirsiniz:

```
$kullaniciAdi = "Ahmet";
```

- Burada, **$kullaniciAdi** değişkeni **"Ahmet"** isimli bir veri parçasını saklar. Bu değişkenin ismi $kullaniciAdi şeklindedir ve $ işareti bu ismin bir değişken olduğunu belirtir. Yani, $ işareti sayesinde PHP, kullaniciAdi kelimesinin bir değişken olduğunu ve bir değer sakladığını anlar.
- PHP'de $ işaretinin kullanılmasının temel nedeni, programın kod içindeki değişkenleri kolayca tanımasına yardımcı olmaktır. Bu sayede, değişkenler ve diğer kod parçaları (fonksiyonlar, sınıflar vb.) arasında bir ayrım yapılır ve kodun okunabilirliği artar.

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

- PHP'de parantezler ve noktalama işaretleri, cümle kurar gibi kod yazmanıza yardımcı olur. Her biri, kodun farklı bölümlerini bir araya getirmek, belirli işlemleri gruplamak veya kodunuzun ne zaman başlayıp bittiğini belirtmek için kullanılır.
- PHP kodu, açılış <?php ve kapanış ?> etiketleri arasına yazılır.
- Kısa açılış ve kapanış etiketi <? .. ?> şeklindedir.

ÖRNEK:

```
<?php
echo ‘Merhaba dünya’
?>
```

Çıktı: Merhaba dünya

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Noktalı Virgül (;)

- PHP'de bir cümleyi bitirdiğinizi belirtir. Her bir komut ya da ifade, genellikle bir noktalı virgülle sonlandırılır. Bu, PHP'ye o satırın bittiğini ve yeni bir satıra geçebileceğini söyler. Örnek:

```
$yas = 15;
```

Burada 15 değerini yas değişkenine atadık ve bu ifadeyi noktalı virgülle sonlandırdık.

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Parantezler ()

- Fonksiyon çağrılarında kullanılır. Bir fonksiyonun adından sonra gelen parantez içine, o fonksiyona vermek istediğiniz bilgiler yazılır. Ayrıca, matematiksel işlemlerde işlem sırasını belirlemek için de kullanılırlar.
- Örnek: **hesapla($yas);**

Burada hesapla fonksiyonuna yas değişkenini veriyoruz.

- Örnek: ($a + $b) * $c;

Burada önce a ve b toplanır, sonra sonuç c ile çarpılır.

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

- **3. Süslü Parantezler {}**
- Kod bloklarını gruplamak için kullanılır. Örneğin, bir if koşulunun ya da bir döngünün (for, while) hangi kodları içerdiğini belirlemek için süslü parantezler kullanılır.

  Örnek:

```
if ($yas > 18) {
echo "Reşitsiniz";
}
```

  Bu örnekte, eğer yas değişkeni 18'den büyükse "Reşitsiniz" yazısını ekrana yazdırır. İfade içindeki kod bloğu süslü parantezler arasına yazılır.

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Köşeli Parantezler []

- Dizilerle çalışırken kullanılır. Bir dizi içindeki belirli bir elemana erişmek veya bir diziye eleman eklemek için köşeli parantezler kullanılır.

Örnek:

```
$isimler = ["Ahmet", "Mehmet"];
echo $isimler[0];
```

Burada isimler dizisinin ilk elemanı ("Ahmet") ekrana yazdırılır.

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Nokta (.)

- PHP'de iki metni (string) birleştirmek için kullanılır. Bu işlem, birleştirme olarak bilinir.

Örnek:

```
$tamIsim = $isim . " " . $soyisim;
```

Burada, isim ve soyisim değişkenleri arasına boşluk koyarak birleştirilir ve yeni bir metin oluşturulur.

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Ters Taksim () Kullanımı: PHP'de en basit escaping yöntemi, özel karakterlerin önüne ters taksim (\) koymaktır. Bu, özellikle çift tırnak (") ve tek tırnak (') kullanımında yaygındır.

```
$ornek = "Onur\'un sözü şu oldu: \"Merhaba!\" ";
echo $ornek;
```

Çıktı: Onur'un sözü şu oldu: "Merhaba!"

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

=> karakteri: PHP'de => işareti, genellikle dizi elemanlarını anahtar-değer çiftleri olarak ilişkilendirmek için kullanılır. Bu işaret, dizi içindeki bir anahtarı (key) karşılık gelen değeri (value) ile eşleştirir. Bir dizi tanımlarken, => işaretinin sol tarafına anahtar (genellikle bir string veya integer), sağ tarafına ise o anahtara ait değer yazılır.

```
$mevye_renkleri = array(
    "elma" => "kırmızı",
    "muz" => "sarı",
    "üzüm" => "mor"
);
```

Bu örnekte, "elma", "muz" ve "üzüm" dizinin anahtarlarıdır ve bunlar sırasıyla "kırmızı", "sarı" ve "mor" değerleriyle eşleştirilmiştir. Bu yapı sayesinde her meyvenin rengini kolaylıkla alabiliriz:

```
echo $mevye_renkleri["elma"]; // "kırmızı" yazdırır
```

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Girintileme ve Boşluk Kullanımı: Kodun okunabilirliğini artırmak için girintileme ve uygun boşluk kullanımı önemlidir.

```
<?php
function checkAge($age) {
    if ($age >= 18) {
        echo "Yetişkin";
    } else {
        echo "Reşit değil";
    }
}
?>
```

2.2. B) KOD YAZIMINDA PARANTEZLERİN VE NOKTALAMA İŞARETLERİNİN NERELERDE VE NASIL KULLANILACAĞI AÇIKLANIR.

Tek ve Çok Satırlık Yorumlar: Kodun anlaşılabilirliğini artırmak için yorumlar kullanılabilir.

```
<?php

// Bu bir tek satırlık yorumdur

# Bu da bir başka tek satırlık yorumdur


/* Bu bir çok satırlık

  yorum bloğudur

  ve birden fazla satırı kapsayabilir */

?>
```

2.2. C) DEĞİŞKEN TANIMLAMA KURALLARI AÇIKLANIR.

PHP'de değişken tanımlama, verileri saklamak ve kod içinde bu verilere erişmek için kullanılır. Değişkenler, **$** işareti ile başlar ve ardından değişkenin adı gelir. PHP'de değişken tanımlarken uymanız gereken bazı temel kurallar ve pratikler vardır:

Değişken İsimlendirme Kuralları:

- **Başlangıç**: Değişken isimleri bir harf (A-Z ya da a-z) veya alt çizgi (_) ile başlamalıdır.
- **Karakterler**: Değişken isimlerinde harfler, rakamlar (0-9) ve alt çizgiler (_) kullanılabilir.
- **Büyük/Küçük Harf Duyarlılığı**: PHP değişken isimlerinde büyük/küçük harfe duyarlıdır. Yani $yas ve $Yas PHP'de iki farklı değişkendir.
- **Özel Karakterler**: Değişken isimlerinde boşluklar, noktalama işaretleri ve özel karakterler kullanılamaz.

2.2. C) DEĞİŞKEN TANIMLAMA KURALLARI AÇIKLANIR.

Pratik Kurallar:

- **Anlamlı İsimler**: Değişkenlerinize, içerdikleri veriyi açıkça yansıtan isimler verin. Örneğin, kullanıcının yaşını saklamak için $yas ya da bir kullanıcının adını saklamak için $kullaniciAdi gibi.
- **Okunabilirlik**: Okunabilirliği artırmak için, birden fazla kelime içeren değişken isimlerinde camelCase ($kullaniciAdi) veya snake_case ($kullanici_adi) kullanılabilir.
- **Kısaltmalardan Kaçının**: Mümkün olduğunca kısaltma kullanmaktan kaçının. Değişken isimlerinin açık ve anlaşılır olması önemlidir.

2.2. C) DEĞİŞKEN TANIMLAMA KURALLARI AÇIKLANIR.

Örnekler:

```
$ad = "Ahmet";          // Geçerli
$_soyad = "Yılmaz";     // Geçerli
$3yas = 25;             // Geçersiz, rakamla başlayamaz
$kul-lanici = "ayse";   // Geçersiz, özel karakter içeriyor
```

PHP'de değişken tanımlarken bu kurallara ve pratiklere uyulması, kodun daha temiz, anlaşılır ve bakımı kolay olmasını sağlar.

2.2. Ç) VERİ TÜRLERİ AÇIKLANIR.

PHP, farklı türde verileri işlemek için çeşitli veri türlerini destekler. Bu veri türleri, değişkenlerin içerebileceği değerlerin türünü belirler ve PHP'nin zengin bir dille olmasını sağlar.

2.2. Ç) VERİ TÜRLERİ AÇIKLANIR.

Boolean: Bu, evet veya hayır gibi iki değerden birini alır. Mesela, bir soruya "evet" veya "hayır" ile cevap vermek gibi düşünebilirsin. Doğru (TRUE) veya yanlış (FALSE) değerlerini saklar. Genellikle koşullu ifadelerde kullanılır.

```
$dogru = TRUE;      // veya $dogru = true;
$yanlis = FALSE;    // veya $yanlis = false;
```

Integer: Tam sayıları saklar. Negatif veya pozitif tam sayılar olabilir. Örneğin, yaşınız, sınıftaki öğrenci sayısı gibi sayılar tam sayıdır.

```
$yas = 21;
$miktar = 0;
$sicaklik = -10;
```

2.2. Ç) VERİ TÜRLERİ AÇIKLANIR.

Float (veya double): Ondalık sayıları saklar. Daha büyük veya hassas sayısal değerler için kullanılır.

$fiyat = 10.99;

$ortalama = 3.14;

String: Metin, kelime veya karakteri ifade eder. Tek tırnak (') veya çift tırnak (") ile çevrili olabilir. Adınız, adresiniz veya en sevdiğiniz kitabın adı birer metin örneğidir.

$isim = 'Duru';

$cumle = "Merhaba, nasılsınız? ";

2.2. Ç) VERİ TÜRLERİ AÇIKLANIR.

Array: Bir dizi şeyi bir arada tutmanıza olanak tanır. Mesela, sınıftaki arkadaşlarınızın isimlerini bir listeye yazmak gibi düşünebilirsiniz. Bu liste, her bir öğrencinin ismini bir sırayla tutar.

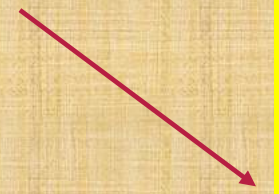
```
$meyveler = array("Azra", "Defne", "Giray");

// Veya PHP 5.4 ve sonrası için kısa dizi sözdizimi

$sebzeler = ["Patetes", "Havuç", "Soğan"];
```

Object (Nesne): Nesneleri saklar. PHP'de sınıflardan türetilen nesneler, özellikler (değişkenler) ve yöntemler (fonksiyonlar) içerebilir. Nesneler, daha karmaşık bilgileri saklamak için kullanılır. Bir telefonun markası, modeli, rengi gibi özellikleri bir arada tutmak istediğinizde bir nesne kullanabilirsiniz.

Object (Nesne) örneği:

```
class Araba {
    public $renk;
    public $model;

    public function __construct($model, $renk) {
        $this->model = $model;
        $this->renk = $renk;
    }

    public function detaylar() {
        return "Bu bir $this->renk renkli $this->model.";
    }
}

$arabam = new Araba("Toyota", "Kırmızı");
echo $arabam->detaylar();
```

Object (Nesne)



2.2. Ç) VERİ TÜRLERİ AÇIKLANIR.

NULL: Hiçbir şeyi ifade etmez. Bir kutunun boş olduğunu düşünün; işte o kutu NULL'dır.

Örnek:

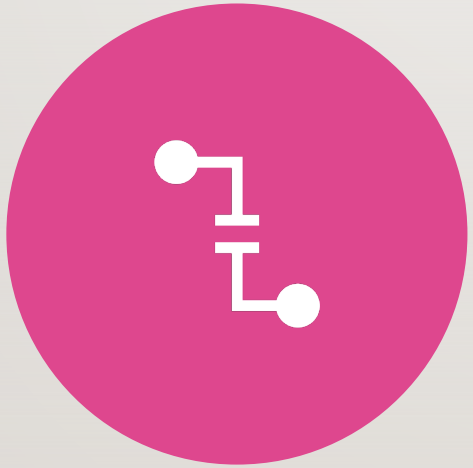
```
$butce = NULL;
```

Not: PHP'de büyük/küçük harf duyarlılığı olmadığı için NULL, null veya Null olarak yazılabilir.

2.2. Ç) VERİ TÜRLERİ AÇIKLANIR.

- PHP'de değişkenlere başlangıçta bir tür ataması yapmanız gerekmez;
- Değişkenin türü, ona atanan değere göre otomatik olarak belirlenir.

2.3. YAPILANDIRMA AYARLARINI UYGULAR.



A) PHP.İNİ DOSYASINDAKİ AYARLAR AÇIKLANIR.

B) ÖĞRENCİLERDEN UYGULAMALI OLARAK PHP.İNİ DOSYASINDA FARKLI AYARLAR YAPMALARI İSTENİR.

2.3. A) PHP.İNİ DOSYASINDAKİ AYARLAR AÇIKLANIR.

PHP.ini dosyası, PHP'nin çalışma şeklini ayarlayan bir yapılandırma dosyasıdır. Bu dosyada birçok ayar bulunur ve PHP'nin performansını, güvenliğini ve davranışını etkileyebilir.

PHP.ini dosyasının yerini bulmak için, PHP yüklü bir sunucuda veya bilgisayarda komut satırını açıp php --ini komutunu çalıştırabilirsiniz. Bu komut, PHP.ini dosyasının konumunu gösterir.

PHP.ini dosyasını düzenlemek için, bir metin editörü kullanabilirsiniz. Windows'ta Notepad veya Notepad++, macOS'ta TextEdit veya başka bir metin editörü işinizi görecektir. Dosyayı açtıktan sonra, değiştirmek istediğiniz ayarları bulup düzenleyebilirsiniz.

2.3. A) PHP.İNİ DOSYASINDAKİ AYARLAR AÇIKLANIR.

Temel Ayarlar

- **short_open_tag**: PHP açılış etiketlerinin kısa versiyonu (<? yerine <?php) kullanılıp kullanılmayacağını belirler.
- **precision**: Kayan noktalı sayıların gösterimindeki maksimum basamak sayısını belirler.

Hata Ayarları

- **display_errors**: Hataların kullanıcıya gösterilip gösterilmeyeceğini belirler.
- **log_errors**: Hataların bir dosyaya kaydedilip kaydedilmeyeceğini belirler.
- **error_reporting**: Hangi tür hataların rapor edileceğini belirler.

Dosya Yükleme Ayarları

- **file_uploads**: PHP üzerinden dosya yükleme işleminin etkin olup olmadığını belirler.
- **upload_max_filesize**: PHP üzerinden yüklenebilecek maksimum dosya boyutunu belirler.
- **max_file_uploads**: Bir istekte yüklenebilecek maksimum dosya sayısını belirler.

2.3. A) PHP.İNİ DOSYASINDAKİ AYARLAR AÇIKLANIR.

Zaman Aşımı ve Bellek Ayarları

- **max_execution_time**: Bir scriptin maksimum çalışma süresini saniye olarak belirler.
- **max_input_time**: Bir scriptin veri almak için harcayabileceği maksimum süreyi belirler.
- **memory_limit**: Bir scriptin kullanabileceği maksimum bellek miktarını belirler.

Veritabanı ve Ağ Ayarları

- **pdo_mysql.default_socket**: PDO_MySQL uzantısı için MySQL soketinin varsayılan yolu.
- **mysqli.default_port**: MySQLi uzantısı için varsayılan port numarası.

2.3. A) PHP.İNİ DOSYASINDAKİ AYARLAR AÇIKLANIR.

Oturum (Session) Ayarları

- **session.save_handler**: Oturum bilgilerinin nasıl saklanacağını belirler (örneğin, dosyalar veya veritabanları).
- **session.save_path**: Oturum dosyalarının saklanacağı yol.
- **session.use_cookies**: Oturum kimliğinin cookie'ler aracılığıyla yönetilip yönetilmeyeceğini belirler.
- **session.use_trans_sid**: Şeffaf oturum kimliği yönetiminin etkin olup olmadığını belirler. Bu, cookie desteklemeyen tarayıcılarda oturum kimliğini URL üzerinden taşımak için kullanılabilir.

Güvenlik Ayarları

- **expose_php**: PHP'nin kendisini ve sürümünü dışa vurup vurmayacağını belirler.
- **disable_functions**: Çalıştırılması yasaklanmış olan PHP fonksiyonlarının listesini belirler.
- **open_basedir**: PHP'nin erişebileceği dosya ve dizinlerin kökünü sınırlar.

2.3. A) PHP.İNİ DOSYASINDAKİ AYARLAR AÇIKLANIR.

Değişiklikleri Uygulama

- PHP.ini dosyasında yaptığınız değişikliklerin etkili olabilmesi için, PHP'yi (genellikle web sunucunuzu) yeniden başlatmanız gerekebilir.

İpucu

- PHP.ini dosyasını düzenlerken, orijinal dosyanın bir yedeğini almak iyi bir fikirdir. Böylece bir sorunla karşılaşıldığında orijinal ayarlara kolayca geri dönülebilir.

2.3. B) ÖĞRENCİLERDEN UYGULAMALI OLARAK PHP.İNİ DOSYASINDA FARKLI AYARLAR YAPMALARI İSTENİR.

Öğrenciye değişikliği hemen görülebilmesi adına short_open_tag ayarı yaptırılabilir.

index.php dosyasında ilk önce aşağıdaki kod yazılır

```
<?
print("Merhaba");
?>
```

Bu kodların çalışmadığı görülür. Öğrenciye php.ini dosyası açtırılır.

```
short_open_tag = Off
//Bu ayar aşağıdaki şekilde değiştirilir.
short_open_tag = On
```

Aynı kodlar tekrar çalıştırılır ve tarayıcıda Merhaba yazısı görülür.

3. ÜNİTE: PHP FONKSİYONLARI, ARİTMETİKSEL VE MANTIKSAL İŞLEMLER

Ünite Açıklaması

- Bu ünitede, PHP programlama dilinde kullanılan matematiksel ve metinsel fonksiyonların kullanımı, koşul yapılarıyla mantıksal ve aritmetiksel işlemler, döngü yapılarının çalışma şekilleri konularına değinilmiştir.

Değerler

- Öz denetim (Kazanım 3.1., 3.2.)
- Sorumluluk (Kazanım 3.1., 3.2., 3.3.)

Alan Becerileri

- Genelleme Becerisi (Kazanım 3.1., 3.2.)
- Kodlama, Programlama Becerisi (Kazanım 3.2., 3.3.)

3.1. PHP PROGRAMLAMA DİLİ FONKSİYONLARINI TANIR.

a) Fonksiyonların işlevi anlatılır.

b) Fonksiyonların kullanım gereksinimi nedenleriyle açıklanır.

c) Fonksiyon kullanarak ve fonksiyon kullanmadan kod yazım örnekleri verilerek fonksiyonların avantajlarına değinilir.

ç) Matematiksel ve metinsel fonksiyonlar anlatılır.

3.1. A) FONKSİYONLARIN İŞLEVİ ANLATILIR.

Fonksiyon Nedir? Hayal et, içine bir şey koyduğun sihirli bir kutun var ve bu kutu bir şeyler yapıyor ve sana bir şey veriyor. İşte PHP'deki fonksiyonlar da tam olarak böyle çalışır.

Fonksiyonlara Neden İhtiyacımız Var? Fonksiyonlar, büyük bir program içindeki küçük programlar gibidir. Kodumuzu daha iyi düzenlememize yardımcı olur. Aynı kodu defalarca tekrar yazmak yerine, onu bir fonksiyona koyabilir ve ihtiyacımız olduğunda kullanabiliriz.

3.1. A) FONKSİYONLARIN İŞLEVİ ANLATILIR.

Fonksiyonlar Nasıl Çalışır? Bir fonksiyonun bir ismi vardır, mesela "toplaSayılar()" veya "alanHesapla()". Bir fonksiyonu kullanmak istediğinizde, adını çağırırsınız ve ona bazı girdiler verirsiniZ. Fonksiyon daha sonra işini yapar ve size bir çıktı verir.

Fonksiyonlar Ne İşe Yarar? Fonksiyonlar her türlü işi yapabilir. Sayıları toplayabilir, bir şifrenin doğru olup olmadığını kontrol edebilir veya hatta yeni bir resim oluşturabilir. Temelde, programınızın yapmasını istediğiniz herhangi bir şeyi düşünebiliyorsanız, muhtemelen bunun için bir fonksiyon yazabilirsiniz.

3.1. B) FONKSİYONLARIN KULLANIM GEREKSİNİMİ NEDENLERİYLE AÇIKLANIR.

1. **Daha Az Kod Yazmak İçin:** Fonksiyonlar, aynı kodu tekrar tekrar yazmaktan kaçınmanıza yardımcı olur. Örneğin, bir web sitesindeki tüm sayfaların başlığını değiştirmek istiyorsanız, bunu her sayfa için ayrı ayrı yapmak yerine bir fonksiyon oluşturabilir ve her sayfada bu fonksiyonu çağırabilirsiniz.
2. **Kodunuzu Daha Kolay Anlaşılır Hale Getirir:** Fonksiyonlar, kodunuzu daha modüler hale getirir. Yani, kodunuzda bir sorun olduğunda, hatayı bulmak ve düzeltmek daha kolay olur. Çünkü her işlevin (fonksiyonun) belirli bir görevi vardır ve bu görevi yerine getirirken diğer kod parçalarından bağımsızdır.

3.1. B) FONKSİYONLARIN KULLANIM GEREKSİNİMİ NEDENLERİYLE AÇIKLANIR.

3. **Kodunuzu Daha Düzenli Hale Getirir:** Fonksiyonlar, kodunuzu daha düzenli hale getirir. Özellikle büyük projelerde, kodunuzda kaybolmamak ve düzeni korumak önemlidir. Fonksiyonlar, kodunuzu daha organize bir şekilde tutmanıza yardımcı olur.
4. **Tekrar Kullanılabilirlik:** Bir fonksiyonu bir kez yazdıktan sonra, ihtiyacınız olduğunda tekrar tekrar kullanabilirsiniz. Bu, zaman kazanmanıza ve daha verimli bir şekilde çalışmanıza olanak tanır.

3.1. B) FONKSİYONLARIN KULLANIM GEREKSİNİMİ NEDENLERİYLE AÇIKLANIR.

1. **Hata Ayıklamayı Kolaylaştırır:** Fonksiyonlar, kodunuzu parçalara ayırdığı için hata ayıklamayı kolaylaştırır. Her fonksiyonun belirli bir amacı olduğu için, hataları bulmak ve düzeltmek daha kolaydır.
2. **Kodunuzun Daha Taşınabilir Olmasını Sağlar:** Bir fonksiyonu bir kez yazdıktan sonra, farklı projelerde veya farklı dosyalarda kolayca kullanabilirsiniz. Bu, kodunuzu daha taşınabilir hale getirir ve tekrar kullanılabilirliği artırır.

3.1. C) FONKSİYON KULLANARAK VE FONKSİYON KULLANMADAN KOD YAZIM ÖRNEKLERİ VERİLEREK FONKSİYONLARIN AVANTAJLARINA DEĞİNİLİR.

Bir dizedeki harf sayısını hesaplayan bir fonksiyon yazalım.

```
function harfSayisi($dize) {
    return strlen($dize);
}
$dize = "Merhaba Dünya";
echo "Dizedeki harf sayısı: " . harfSayisi($dize);
```

3.1. C) FONKSİYON KULLANARAK VE FONKSİYON KULLANMADAN KOD YAZIM ÖRNEKLERİ VERİLEREK FONKSİYONLARIN AVANTAJLARINA DEĞİNİLİR.

- **Fonksiyon kullanmadan** örnek bir dizedeki harf sayısını doğrudan hesaplayalım.

```
$dize = "Merhaba Dünya";

$harfSayısı = strlen($dize);

echo "Dizedeki harf sayısı: " . $harfSayısı;
```

FONKSİYON YAZIMI

```
$dize = "Merhaba Dünya";
echo "Dizedeki harf sayısı: " . harfSayisi($dize);
```

FONKSİYON YAZILMADAN

```
$dize = "Merhaba Dünya";
$harfSayısı = strlen($dize);
echo "Dizedeki harf sayısı: " . $harfSayısı;
```

FONKSİYON YAZMANIN AVANTAJLARI

Daha Az Kod Yazma
Daha Modüler Kod
Tekrar Kullanılabilirlik

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Matematiksel Fonksiyonlar:

Toplama ve Çıkarma: Matematiksel fonksiyonlar, sayılarla çalışır ve genellikle işlemleri gerçekleştirmek için kullanılır. Örneğin, toplama ve çıkarma işlemleri yapmak için kullanılabilirler.

```
$toplam = 5 + 3;

$fark = 7 - 2;
```

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Matematiksel Fonksiyonlar:

Çarpma ve Bölme: Çarpma ve bölme işlemleri için de matematiksel fonksiyonlar kullanılabilir.

```
$carpim = 4 * 6;

$bolum = 10 / 2;
```

3.1. Ç)
MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Matematiksel Fonksiyonlar:

Dizilerden minumum ve maksimum sayı elde etme:

```
echo(min(0, 150, 30, 20, -8, -200) ;
```

Çıktı: -200

```
echo(max(0, 150, 30, 20, -8, -200));
```

Çıktı: 150

Pi sayısını elde etme:

```
echo(pi());
```

Çıktı: 3.1415926535898

3.1. Ç)
MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Matematiksel Fonksiyonlar:

Karekök alma:

```
echo(sqrt(64));
```

Çıktı: 8

Mutlak değer alma:

```
echo(abs(-100));
```

Çıktı: 100

Yuvarlama fonksiyonu:

```
echo(round(0.60);
```
Çıktı: 1

```
echo(round(0.50);
```
Çıktı: 1

```
echo(round(0.49);
```
Çıktı: 0

3.1. Ç)
MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Matematiksel Fonksiyonlar:

Rastgele tamsayı üretme

```
echo(rand(10, 100));
```

//10 ve 100 arasında rastgele tamsayı üretir.

Çıktı: 74 (Her run işleminde farklı sonuçlar çıkacaktır.)

Tüm matematik fonksiyonları için aşağıdaki web adresine bakılabilir.

https://www.w3schools.com/php/php_ref_math.asp

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Metinsel Fonksiyonlar:

Dize Uzunluğu: Bir dizenin uzunluğunu bulmak için kullanılabilirler.

```
$dize = "Merhaba Dünya";
$uzunluk = strlen($dize);
```

Çıktı: 13

3.1. Ç)
MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Metinsel Fonksiyonlar:

Kelime Uzunluğu: Bir dizenin kelime sayısını bulmak için kullanılabilirler.

```
echo str_word_count("Merhaba PHP");
```

Çıktı: 2

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Metinsel Fonksiyonlar:

Dize Birleştirme: İki veya daha fazla diziyi birleştirmek için metinsel fonksiyonlar kullanılabilir.

```
$ad = "Sezer";
$soyad = "Yılmaz";
$tamAd = $ad . " " . $soyad;
echo ($tamAd)
```

Çıktı: Sezer Yılmaz

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Metinsel Fonksiyonlar:

strpos() Fonksiyonu

- strpos() fonksiyonuna ilk parametre olarak gönderilen string içerisinde ikinci parametre olarak gönderilen string aranır. Eğer string veri bulunursa aranan string verinin ilk karakterinin indis numarası döndürülür eğer bulamazsa False bilgi gönderilir.

```
echo strpos("Merhaba PHP", "PHP");
```

Çıktı: 8

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Metinsel Fonksiyonlar:

str_replace() Fonksiyonu

- str_replace() fonksiyonu ile bir string içerisinde istenen karakter ya da karakterler güncellenir.

```
echo str_replace("PHP", "Javascript", "Hello PHP");
```

"Hello PHP" içerisinde PHP karakterleri aranır ve yerine Javascript bilgisi yazdırılır.

Çıktı: Hello Javascript

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Metinsel Fonksiyonlar:

substr() Fonksiyonu

- substr() fonksiyonu ile bir string verinin istenilen bölümü alınır.

```
echo substr("Merhaba PHP", 8);
```

8.karakterden itibaren tüm karakterleri alır ve ekrana **"PHP"** yazar.

- Substr() fonksiyonuna gönderilen 3.parametreyle kaç karakter seçileceğini belirtebiliriz.

```
echo substr("Merhaba PHP", 0, 7);
```

0. karakterden başlar 7 karakter alır ve ekrana **"Merhaba"** yazar.

3.1. Ç) MATEMATİKSEL VE METİNSEL FONKSİYONLAR ANLATILIR.

Tüm metinsel fonksiyonlar için aşağıdaki web adresine bakılabilir.

https://www.w3schools.com/php/php_ref_string.asp

Youtube kanal önerisi: Fehmi Uyar

https://www.youtube.com/watch?v=CPcpcqaQRvw&list=PLY20HpFruiK2RxKOaKIirxjlYAA3dB6J6

3.2. PHP PROGRAMLAMA DİLİNDE MANTIKSAL VE ARİTMETİKSEL İŞLEMLERİ KAVRAR.

- a) Karar yapıları anlatılır.
- b) Mantıksal ve aritmetiksel operatörler açıklanır.
- c) Örnekler verilerek hangi operatörün kullanılacağı sorulur.

3.2.
A) KARAR YAPILARI ANLATILIR.

- Karar yapıları, programların belirli koşullara göre farklı yollar izlemesini sağlar. Yani, bir karar alır ve buna göre hareket ederler. Bu, programların esnek ve dinamik olmasını sağlar.

3.2.
A) KARAR YAPILARI ANLATILIR.

Karşılaştırma operatörleri:

Eşitlik (==)

- İki değerin eşit olup olmadığını kontrol eder. Değerler eşitse TRUE döner.
- if ($a == $b) // $a ve $b değer olarak eşitse

Özdeşlik (===)

- İki değerin hem değer olarak hem de tür olarak eşit olup olmadığını kontrol eder. Her ikisi de eşitse TRUE döner.

if ($a === $b) // $a ve $b değer ve tür olarak eşitse

Eşit Değil (!= veya <>)

- İki değerin eşit olmadığını kontrol eder. Değerler eşit değilse TRUE döner.
- if ($a != $b) // $a ve $b değer olarak eşit değilse

Özdeş Olmayan (!==)

- İki değerin hem değer olarak hem de tür olarak eşit olmadığını kontrol eder. Her ikisi de eşit değilse TRUE döner.
- if ($a !== $b) // $a ve $b hem değer hem de tür olarak eşit değilse

3.2.
A) KARAR YAPILARI ANLATILIR.

Karşılaştırma operatörleri:

Büyüktür (>)

- Bir değerin diğer değerden büyük olup olmadığını kontrol eder.
- if ($a > $b) // $a, $b'den büyükse

Küçüktür (<)

- Bir değerin diğer değerden küçük olup olmadığını kontrol eder.
- if ($a < $b) // $a, $b'den küçükse

Büyük Eşittir (>=)

- Bir değerin diğer değerden büyük veya ona eşit olup olmadığını kontrol eder.
- if ($a >= $b) // $a, $b'den büyük veya ona eşitse

Küçük Eşittir (<=)

- Bir değerin diğer değerden küçük veya ona eşit olup olmadığını kontrol eder.
- if ($a <= $b) // $a, $b'den küçük veya ona eşitse

3.2.
A) KARAR YAPILARI ANLATILIR.

If-Else Yapısı: If-else yapısı, belirli bir koşulun doğru veya yanlış olmasına bağlı olarak farklı kod bloklarının çalıştırılmasını sağlar. Örneğin, bir öğrencinin sınav notuna göre başarılı veya başarısız olduğunu belirleyebiliriz:

```
$sınav_notu = 75;
if ($sınav_notu >= 50) {
    echo "Tebrikler, sınavı geçtin!";
} else {
    echo "Üzgünüm, sınavı geçemedin.";
}
```

3.2.
A) KARAR YAPILARI ANLATILIR.

Switch-Case Yapısı: Switch-case yapısı, bir değişkenin farklı değerlerine göre farklı kod bloklarının çalıştırılmasını sağlar. Örneğin, bir ayın kaç gün olduğunu belirleyebiliriz:

```
$ay = "Şubat";
switch ($ay) {
    case "Nisan":
    case "Haziran":
    case "Eylül":
    case "Kasım":
        echo "Bu ay 30 gün.";
        break;
    case "Şubat":
        echo "Bu ay 28 veya 29 gün.";
        break;
    default:
        echo "Bu ay 31 gün.";
        break;
}
```

3.2.

B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Aritmetik Operatörler

- Aritmetik operatörler, matematikteki temel işlemleri (toplama, çıkarma, çarpma, bölme ve mod alma) gerçekleştirmek için kullanılır. Bunlar günlük hayattaki basit hesaplamalarla aynıdır.
- **Toplama (+)**: İki sayıyı toplar. Örneğin, 3 + 4 ifadesi 7 sonucunu verir.
- **Çıkarma (-)**: İlk sayıdan ikinci sayıyı çıkarır. Örneğin, 5 - 2 ifadesi 3 sonucunu verir.
- **Çarpma (*)**: İki sayıyı çarpar. Örneğin, 6 * 7 ifadesi 42 sonucunu verir.
- **Bölme (/)**: İlk sayıyı ikinci sayıya böler. Örneğin, 8 / 4 ifadesi 2 sonucunu verir.
- **Mod (%)**: İlk sayının ikinci sayıya bölünmesiyle kalanı verir. Örneğin, 10 % 3 ifadesi 1 sonucunu verir, çünkü 10'u 3'e böldüğünüzde kalan 1'dir.

3.2.
B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Aritmetik Operatörler

Artırma Operatörleri

- **Ön Artırma (++$x):** Değişkenin değerini bir artırır ve ardından değişkenin yeni değerini döndürür. Örnek:

```
$x = 5;

echo ++$x;          // 6'yı yazdırır ve $x'in değeri 6 olur
```

- **Sonra Artırma ($x++):** Değişkenin mevcut değerini döndürür ve ardından değişkenin değerini bir artırır.

```
$x = 5;

echo $x++;          // 5'i yazdırır ve ardından $x'in değerini 6 yapar

echo $x;            // Sonraki kullanımda $x'in değeri 6 olur
```

* Ön azaltma işleminde -- operatörü kullanılır.

3.2.

B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Aritmetik Operatörler

- `$toplam = 3 + 4;` // 7 değerini verir
- `$fark = 5 - 2;` // 3 değerini verir
- `$carpim = 6 * 7;` // 42 değerini verir
- `$bolum = 8 / 4;` // 2 değerini verir
- `$mod = 10 % 3;` // 1 değerini verir

3.2.
B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Aritmetik Operatörler

- PHP'de aritmetik işlemler yaparken kısaltmaları kullanmak, kodunuzu daha okunabilir ve yazımı daha hızlı hale getirebilir. Bu kısaltmalar, bir değişkenin mevcut değeri üzerinde doğrudan işlem yapmanıza ve sonucu aynı değişkene atamanıza olanak tanır.

Toplama ve Atama (+=)

- Bir değişkene bir değer ekler ve sonucu aynı değişkene atar.

```
$x = 5;

$x += 3;          // $x = $x + 3 ile aynı
```

Çıktı: 8

Çıkarma ve Atama (-=)

- Bir değişkenden bir değer çıkarır ve sonucu aynı değişkene atar.

```
$x = 5;

$x -= 2;          $x = $x - 2 ile aynı
```

Çıktı: 3

3.2.
B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Aritmetik Operatörler

- PHP'de aritmetik işlemler yaparken kısaltmaları kullanmak, kodunuzu daha okunabilir ve yazımı daha hızlı hale getirebilir. Bu kısaltmalar, bir değişkenin mevcut değeri üzerinde doğrudan işlem yapmanıza ve sonucu aynı değişkene atamanıza olanak tanır.

Çarpma ve Atama(*=)

- Bir değişkeni bir değerle çarpar ve sonucu aynı değişkene atar.

```
$x = 6;

$x *= 4;        // $x = $x * 4 ile aynı
```

Çıktı: 24

Bölme ve Atama (/=)

- Bir değişkeni bir değere böler ve sonucu aynı değişkene atar.

```
$x = 20;

$x /= 5;        $x = $x / 5 ile aynı
```

Çıktı: 4

3.2.

B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Aritmetik Operatörler

- PHP'de aritmetik işlemler yaparken kısaltmaları kullanmak, kodunuzu daha okunabilir ve yazımı daha hızlı hale getirebilir. Bu kısaltmalar, bir değişkenin mevcut değeri üzerinde doğrudan işlem yapmanıza ve sonucu aynı değişkene atamanıza olanak tanır.

Mod ve Atama (/=)

- Bir değişkeni bir değere böler ve kalanı aynı değişkene atar.

```
$x = 20;

$x /= 6;        $x = $x / 6 ile aynı
```

Çıktı: 2

3.2.

B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Mantıksal Operatörler

Mantıksal operatörler, genellikle koşullu ifadelerde (doğru ya da yanlış değerlerle çalışan durumlar) kullanılır. Bu operatörler, verilen koşulların doğru (TRUE) ya da yanlış (FALSE) olup olmadığını kontrol etmek için kullanılır.

- **(AND, &&)**: İki koşulun da doğru olması durumunda TRUE sonucu verir.

Örneğin, eğer x > 5 AND x < 10 koşulları her ikisi de doğruysa, sonuç TRUE olur.

- **(OR, ||)**: İki koşuldan birinin veya her ikisinin de doğru olması durumunda TRUE sonucu verir.

Örneğin, eğer x == 5 OR x == 10 koşullarından biri doğruysa, sonuç TRUE olur.

- **(NOT, !)**: Bir koşulun tersini alır. Eğer koşul doğruysa FALSE, yanlışsa TRUE sonucu verir.

Örneğin, !($x == 5) ifadesi, $x 5'e eşit değilse TRUE, eşitse FALSE sonucunu verir.

3.2.
B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Mantıksal Operatörler

- **Boş Olmayan (Null Coalescing) Operatör**

Bir değişkenin değerini kontrol edip; eğer null ise varsayılan bir değer atar.

Örnek:

```
$kullaniciIsim = null;

$isim = $kullaniciIsim ?? "Misafir";

echo $isim
```

Çıktı: Misafir

3.2.
B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Mantıksal Operatörler

Kullanım Örnekleri

Diyelim ki bir sinema bileti 15 yaş altı ve 65 yaş üstü için indirimlidir. Bu durumu kontrol etmek için mantıksal ve aritmetik operatörleri birleştirebiliriz:

```
$yas = 14;
if ($yas < 15 || $yas > 65) {
    echo "Bilet indirimli.";
} else {
    echo "Bilet tam fiyat.";
}
```

Bu örnekte, kullanıcının yaşı 14 olduğu için "Bilet indirimli." mesajı yazdırılır. Çünkü $yas < 15 koşulu TRUE (doğru) döner ve || (VEYA) operatörü sayesinde sadece bir koşulun TRUE olması yeterlidir.

3.2.
B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

Mantıksal Operatörler

Koşullu (ternary) operatörü:
PHP'de koşullu (ternary) operatörü, basit if-else ifadelerinin kısa bir şekilde yazılmasına olanak tanır. Bu operatör, üç bölümden oluşur: bir koşul, bir soru işareti (?), koşul doğruysa gerçekleşecek ifade, bir iki nokta üst üste (:), ve koşul yanlışsa gerçekleşecek ifade. Koşullu operatörün genel yapısı şöyledir:

```
koşul ? ifade1 : ifade2;
```

Örnek:

Diyelim ki bir kullanıcının yaşına bağlı olarak "yetişkin" veya "çocuk" etiketini atamak istiyorsunuz. Bu durumu koşullu operatör ile şu şekilde yazabilirsiniz:

```
$yas = 20;

$etiket = $yas >= 18 ? "yetişkin" : "çocuk";

echo $etiket;
```

Çıktı: Yetişkin

3.2.

B) MANTIKSAL VE ARİTMETİKSEL OPERATÖRLER AÇIKLANIR.

İşlem Önceliği:

- **Parantez ( ) en yüksek önceliğe sahiptir:** İfadelerdeki işlemlerin sırasını belirlemek için kullanılır.
- **Artırma/Azaltma (++, --):** Değişkenlerin değerlerini artırma veya azaltma işlemleri yüksek önceliklidir.
- **Aritmetik Operatörler (*, /, %, +, -):** Çarpma, bölme ve mod alma işlemleri toplama ve çıkarmadan önce değerlendirilir.
- **Karşılaştırma Operatörleri (<, <=, >, >=, ==, !=, ===, !==):** Bu operatörler aritmetik operatörlerden sonra değerlendirilir.
- **Mantıksal AND (&&):** Mantıksal operatörler arasında öncelik sırası vardır; AND operatörü OR operatöründen (||) önce değerlendirilir.
- **Mantıksal OR (||):** AND operatöründen sonra değerlendirilir.
- **Atama Operatörleri (=, +=, -=, *=, /=, %= vb.):** Genellikle en düşük önceliğe sahiptirler ve ifadenin en sonunda değerlendirilirler.

3.2. C) ÖRNEKLER VERİLEREK HANGİ OPERATÖRÜN KULLANILACAĞI SORULUR.

Soru: Bir alışveriş listesindeki ürünlerin toplam fiyatını hesaplayınız.

Soru: İki sayının birbirine eşit olup olmadığını kontrol ediniz.

Soru: Bir kullanıcının hem kullanıcı adını hem de şifresini doğru girmesini kontrol ediniz.

Soru: Aşağıdaki değişkenin değerini kısaltma yöntemiyle 5 artırınız.

```
$x = 9
```

Soru: Aşağıdaki değişkenin değerini bir ön artırın.

```
$x = 5
```

Soru: Aşağıdaki php kodunun çıktısını bulunuz.

```
$islem = 6 + (8-3*2) / 2

echo $işlem
```

Soru: «Merhaba» ve «Dünya» kelimelerini birleştiriniz.

Soru: Aşağıdaki php kodunun çıktısını bulunuz.

```
$admin = "Kaan";

$ad= $admin ?? "adsız";

echo $isim;
```

Soru: Aşağıdaki işlemin sonucu 3 ise boşluk bırakılan yere hangi operatörler gelebilir?

```
$a = 10

$b = 7

echo $a - - - - $b
```

3.3. DÖNGÜLERİ KULLANIR.

- a) PHP programlama dilindeki döngü yapılarından bahsedilir.
- b) Aralarındaki farklar açıklanır.
- c) Örnekler verilerek hangi döngü yapısını kullanmalarının daha uygun olacağı sorulur.

3.3. A) PHP PROGRAMLAMA DİLİNDEKİ DÖNGÜ YAPILARINDAN BAHSEDİLİR.

PHP'de, programların belirli koşullar altında kod bloklarını tekrar tekrar çalıştırmasına olanak tanıyan birkaç farklı döngü yapısı bulunur. Bu döngü yapıları, kodunuzda tekrar eden görevleri otomatikleştirmek, dizileri işlemek, veritabanından alınan verileri listelemek gibi birçok durumda kullanılabilir. PHP'deki temel döngü yapıları şunlardır:

1. for Döngüsü

2. while Döngüsü

3. do-while Döngüsü

4. foreach Döngüsü

5. break ve continue

3.3. A) PHP PROGRAMLAMA DİLİNDEKİ DÖNGÜ YAPILARINDAN BAHSEDİLİR.

1. for Döngüsü

Örneğin, 1'den 10'a kadar saymak istiyorsun. Bunu yaparken, nereden başlayacağını (1), nerede duracağını (10) ve her adımda ne kadar ilerleyeceğini (her seferinde 1 artır) belirtirsin.

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

Çıktı: 1 2 3 4 5 6 7 8 9 10

3.3. A) PHP PROGRAMLAMA DİLİNDEKİ DÖNGÜ YAPILARINDAN BAHSEDİLİR.

2. while Döngüsü

Bir oyun oynuyorsun ve "Enerjin bitene kadar devam et" diyor. İşte while döngüsü de buna benzer. Bir koşul doğru olduğu sürece işlem yapmaya devam eder. Örneğin elinde 5 tane fındık var ve her seferinde birini yiyorsun, her fındık yediğinde «Ben fındığı çok seviyorum» diyorsun. Tüm fındıklar bitene kadar bu işlemi sürdürüyorsun.

```
$i = 1;

while ($i <= 5) {

    echo 'Ben fındığı çok seviyorum<br>';

    $i++;

}
```

Çıktı:

Ben fındığı çok seviyorum
Ben fındığı çok seviyorum
Ben fındığı çok seviyorum
Ben fındığı çok seviyorum
Ben fındığı çok seviyorum

3.3. A) PHP PROGRAMLAMA DİLİNDEKİ DÖNGÜ YAPILARINDAN BAHSEDİLİR.

3. do-while Döngüsü

Bu, while döngüsüne benzer ama en az bir kere mutlaka yapılması gereken işler için idealdir. "Ödevini yap, sonra oyun oyna" gibidir. Önce ödevini yaparsın (işlemi yaparsın) ve sonra "Daha ödev var mı?" diye kontrol edersin. Eğer varsa, tekrar yaparsın; yoksa oyun oynamaya geçersin.

```
$i = 1;
do {
    echo 'Ödevini yap';
    $i++;
} while ($i <= 0);
```

Çıktı:

Ödevini yap

*Koşul gerçekleşmemesine rağmen eylemi bir kere gerçekleştirdi.

3.3. A) PHP PROGRAMLAMA DİLİNDEKİ DÖNGÜ YAPILARINDAN BAHSEDİLİR.

4. foreach Döngüsü

Arkadaşlarının isimlerini bir listeye yazdın ve her birinin ismini tek tek okumak istiyorsun. foreach döngüsü, bu tür bir liste üzerinde dolaşmanı sağlar. Her bir arkadaşının ismini alır ve sırayla söyler.

```
$arkadaslar = ["Songül", "Melisa", "Murat "];
foreach ($arkadaslar as $isim) {
    echo $isim . ' ';
}
```

Çıktı: Songül Melisa Murat

3.3. A) PHP PROGRAMLAMA DİLİNDEKİ DÖNGÜ YAPILARINDAN BAHSEDİLİR.

5. break ve continue Döngüsü

Bu ikisi, döngülerdeki joker kartlardır. Diyelim ki, sayı sayıyorsun ama 5'i söylemek istemiyorsun. continue kullanarak 5'i atlayıp 6'dan devam edebilirsin. Ya da, 7'ye geldiğinde birden "Yeter, daha fazla saymak istemiyorum" diyorsun. İşte o zaman break kullanırsın ve döngüyü orada bitirirsin.

```
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        continue;        // 5'i atlar ve döngünün bir sonraki adımına geçer.
    }
    if ($i == 7) {
        break;           // 7'ye gelindiğinde döngüyü sonlandırır.
    }
    echo $i . ' ';
}
```

Çıktı: 1 2 3 4 6

3.3. B) ARALARINDAKİ FARKLAR AÇIKLANIR.

for Döngüsü:

- Belirli sayıda tekrar yapmak istediğinde kullanılır.
- Başlangıç, bitiş ve adım büyüklüğü belirlenir.
- Genellikle dizi uzunluğu gibi sabit bir değerle çalışırken tercih edilir.

while Döngüsü:

- Bir koşul doğru olduğu sürece tekrar eder.
- Koşul döngü gövdesinin başında kontrol edilir.
- Döngüye girmeden önce koşulun doğru olup olmadığı bilinmeyebilir, bu yüzden bazen hiç çalışmayabilir.

do-while Döngüsü:

- while döngüsüne benzer, fakat koşul döngü gövdesinin sonunda kontrol edilir.
- Koşul ne olursa olsun döngü gövdesi en az bir kez çalışır.
- Genellikle kullanıcının girişini alıp bir şeyler yapmak istediğinde ve en az bir kez çalışmasını garantilemek istediğinde kullanılır.

foreach Döngüsü:

- Dizileri veya nesneleri yinelemek için kullanılır.
- Dizi ya da nesnenin her elemanı için döngü bir kez çalışır.
- Dizi elemanları üzerinde çalışırken, elemanın anahtarına ve değerine kolayca ulaşmanı sağlar.

3.3.
C) ÖRNEKLER VERİLEREK HANGİ DÖNGÜ YAPISINI KULLANMALARININ DAHA UYGUN OLACAĞI SORULUR.

Soru: Bir sayı değişkeniniz var ve bu değişken sıfır olmadığı sürece bir işlem yapmak istiyorsunuz. Hangi döngü yapısı kullanılmalıdır?

Soru: Bir dizi içerisindeki her bir ürün için stok kontrolü yapmak ve her bir ürünün adını ve stok miktarını yazdırmak istiyorsunuz. Hangi döngü yapısı kullanılmalıdır?

Soru: 10-90 arası sayıları ekrana yazdırmak istiyorsunuz. Hangi döngü yapısı kullanılmalıdır?

Soru: Bir çocuk, balonları şişirip patlatıyor ve bunu en az bir kez yapmak istiyor. Ancak, bir balon çok fazla şişirilirse patlar ve oyun sona erer. Balonun patlayıp patlamadığını kontrol eden bir sayaç var ve bu sayaç her şişirmede artıyor. Sayaç 10'a ulaştığında balon patlar ve oyun biter. Balon şişirme işlemini yapan ve sayaç 10'a ulaşınca duran bir döngü nasıl yazılır?

Cevap:

```
$sayac = 1; // Sayacı başlat

do {

   echo "Balon şişirildi: " . $sayac . "<br>"; // Balon şişirme işlemi

   $sayac++; // Her döngüde sayacı artır

} while ($sayac <= 10); // Sayacın değeri 10'a ulaşana kadar döngü devam eder

    echo "Balon patladı ve oyun bitti!"; // Sayacın değeri 10'a ulaştığında oyun sona erer
```

4. ÜNİTE: PHP PROGRAMLAMA DİLİ İLE VERİ TABANI İŞLEMLERİ

Ünite Açıklaması

- Bu ünitede, PHP programlama dilinde veri tabanı işlemleri anlatılmıştır.

Değerler

- Sabır (Kazanım 4.1., 4.2., 4.3.)
- Öz Denetim (Kazanım 4.1., 4.2., 4.3.)

Alan Becerileri

- Algoritmik Düşünme Becerisi (Kazanım 4.2.)
- Kodlama, Programlama Becerisi (Kazanım 4.1., 4.2.)
- Kodlama, Programlama Becerisi (Kazanım 4.1.)

4.1. VERİ TABANI BAĞLANTISI YAPAR.

MySQL dilinden bahsedilir.

MySQL, bilgisayarlar için bir tür veritabanı yönetim sistemidir. Bunu, çok miktarda bilgiyi düzenli bir şekilde saklayıp, bu bilgilere hızlıca ulaşmak isteyen büyük bir kütüphane olarak düşünebilirsin. Bilgisayar programları bu kütüphaneye sorular sorar (örneğin, "Bu okulda kaç tane 10. sınıf öğrencisi var?" gibi) ve MySQL de bu sorulara cevap verir.

MySQL'de, bilgiler tablolar halinde saklanır. Bir tablo, Excel'deki bir sayfa gibi düşünülebilir, burada her satır belirli bir bilgiyi (örneğin, bir öğrencinin adı, soyadı, sınıfı gibi) ve her sütun bu bilgilerin çeşitlerini (adı, soyadı, sınıfı gibi kategoriler) temsil eder.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

MySQL dilinden bahsedilir.

Öğrencilerle ilgili bir tablo düşünün:

```
+-------+---------+-------+
| Adı   | Soyadı  | Sınıf |
+-------+---------+-------+
| Ayşe  | Yılmaz  | 10    |
| Ahmet | Kaya    | 11    |
| Elif  | Demir   | 10    |
+-------+---------+-------+
```

Burada her bir satır bir öğrenciyi temsil ediyor ve her bir sütun öğrencinin adı, soyadı ve sınıfını gösteriyor.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

MySQL dilinden bahsedilir.

- Programcılar, MySQL'e özel komutlar yazarak bu tablolarda belirli bilgileri arayabilir, yeni bilgiler ekleyebilir, var olan bilgileri değiştirebilir ya da silerler. Bu komutlara SQL (Structured Query Language - Yapılandırılmış Sorgu Dili) komutları denir ve tüm veritabanı işlemlerini yönetmek için kullanılırlar.

- MySQL, web siteleri, uygulamalar ve diğer birçok teknolojiyle entegre çalışabilen popüler ve güçlü bir veritabanı sistemidir. Örneğin, bir web sitesinde kayıt olduğunuzda, adınız, e-posta adresiniz ve diğer bilgileriniz bir MySQL veritabanına kaydedilir. Daha sonra bu bilgilere tekrar ihtiyaç duyduğunuzda, sistem MySQL veritabanına sorgu göndererek bu bilgileri hızlıca alır ve size gösterir.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

PHP programlama dili ile MySQL dilinin birlikte nasıl çalıştığı anlatılır

- PHP ve MySQL, birlikte çalışarak web sitelerinin veri saklamasını, veri almasını ve veri üzerinde işlem yapmasını sağlarlar. Bu ikiliyi bir görev yapmak için birlikte çalışan iki arkadaş gibi düşünebilirsin. PHP, işi yapacak olan, MySQL ise işin yapılacağı yerdeki bilgileri tutan arkadaştır.

- Örneğin, bir sosyal medya sitesi düşün. Bu sitede, kullanıcılar kendi profillerini oluşturabilir, durum güncellemeleri yapabilir ve arkadaşlarının güncellemelerini görebilirler. İşte burada PHP ve MySQL devreye girer.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

PHP'nin Rolü:

- PHP, kullanıcının web sitesinde yaptığı işlemleri alır ve ne yapılması gerektiğine karar verir. Örneğin, bir kullanıcı yeni bir durum güncellemesi paylaştığında, PHP bu bilgiyi alır ve "Tamam, bu güncellemeyi herkesin görebilmesi için bir yere kaydetmem gerekiyor" der.

MySQL'in Rolü:

- MySQL, PHP'nin verdiği bilgileri saklar. Durum güncellemesi örneğinde, PHP durum güncellemesini MySQL veritabanına gönderir ve MySQL de bu bilgiyi bir tabloya kaydeder. Bu tablo, bütün durum güncellemelerinin listesini tutar.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

PHP programlama dili ile MySQL dilinin birlikte nasıl çalıştığı anlatılır

İşbirliği:

- Bir kullanıcı profil sayfasına gittiğinde, PHP önce MySQL'den "Bu kullanıcının tüm durum güncellemelerini al" diye bir istekte bulunur. MySQL bu isteği yerine getirir ve ilgili verileri PHP'ye geri gönderir. PHP de bu bilgileri kullanarak kullanıcıya gösterilecek web sayfasını oluşturur.

Bu süreç genellikle şu adımları içerir:

1. **Kullanıcı İstekte Bulunur:** Kullanıcı, bir form doldurur veya bir butona tıklar.
2. **PHP İsteği İşler:** PHP, kullanıcıdan gelen isteği alır ve ne yapılacağına karar verir.
3. **MySQL İle İletişim:** PHP, MySQL'e bir SQL sorgusu gönderir. Bu sorgu, veri eklemek, güncellemek, silmek veya almak olabilir.
4. **Veritabanı İşlemi:** MySQL, PHP'den gelen sorguyu işler ve gerekli işlemi yapar (veri saklar, günceller, siler veya seçilen verileri döndürür).
5. **Sonuç PHP'ye Döner:** MySQL, işlem sonucunu PHP'ye geri gönderir.
6. **Kullanıcıya Yanıt:** PHP, alınan verileri işler ve kullanıcıya bir web sayfası olarak sunar.

- Bu süreç, PHP ve MySQL kullanılarak yapılan bir web sitesinin temel işleyişidir. PHP, kullanıcıyla etkileşimde bulunan ve 'ön yüz' işlemleri yürüten kısımken, MySQL arka planda verileri saklayan ve yöneten 'arka yüz' kısmını oluşturur.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

PHP programlama dilinde kullanılan MySQL dili komutları açıklanır.

CREATE TABLE: Yeni bir tablo oluşturmak için kullanılır. Örneğin kullanıcılar adlı bir tablo oluşturmak için:

```
CREATE TABLE kullanıcılar (
    id INT AUTO_INCREMENT PRIMARY KEY,
    adı VARCHAR(50),
    soyadı VARCHAR(50),
    email VARCHAR(100)
);
```

DROP TABLE: Bir tabloyu veritabanından silmek için kullanılır. Örneğin kullanıcılar tablosunu silmek için:

```
DROP TABLE kullanıcılar;
```

4.1. VERİ TABANI BAĞLANTISI YAPAR.

PHP programlama dilinde kullanılan MySQL dili komutları açıklanır.

SELECT: Veritabanından bir veya daha fazla kaydı seçmek için kullanılır.

Örneğin bir kullanıcılar tablosundaki tüm kayıtları getirmek için:

```
SELECT * FROM kullanıcılar;
```

veya sadece belirli sütunları (adı ve email) seçmek için:

```
SELECT adı, email FROM kullanıcılar;
```

INSERT INTO: Veritabanına yeni bir kayıt eklemek için kullanılır. Örneğin kullanıcılar tablosuna yeni bir kullanıcı eklemek için:

```
INSERT INTO kullanıcılar (adı, soyadı, email) VALUES ('Ahmet', 'Yılmaz', 'ahmet@gmail.com’);
```

UPDATE: Veritabanındaki mevcut kayıtları güncellemek için kullanılır. Örneğin bir kullanıcının email adresini güncellemek için:

```
UPDATE kullanıcılar SET email='yeniemail@gmail.com' WHERE adı='Ahmet’;
```

JOIN: İki veya daha fazla tabloyu birleştirmek için kullanılır. İlişkili verileri birleştirip sorgulamak istediğinizde kullanışlıdır.

```
SELECT Orders.OrderID, Customers.CustomerName FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

4.1. VERİ TABANI BAĞLANTISI YAPAR.

Veri tabanı bağlantısı örneklerle açıklanır.

PHP ile MySQL Veritabanı Bağlantısı: PHP'de MySQL veritabanı ile etkileşime geçmek için genellikle mysqli veya PDO (PHP Data Objects) kullanılır. İşte basit bir mysqli bağlantı örneği:

```
$baglanti = new mysqli("sunucu_adresi", "kullanıcı_adı", "parola", "veritabanı_adı");
if ($baglanti->connect_error) {
    die("Bağlantı hatası: " . $baglanti->connect_error);
}
$sql = "SELECT adı, email FROM kullanıcılar";
$result = $baglanti->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "Adı: " . $row["adı"]. " - Email: " . $row["email"]. "<br>";
    }
} else {
    echo "0 sonuç bulundu";
}
$baglanti->close();
```

4.1. VERİ TABANI BAĞLANTISI YAPAR.

Veri tabanı bağlantısı örneklerle açıklanır.

```
$baglanti = new mysqli("sunucu_adresi", "kullanıcı_adı", "parola", "veritabanı_adı");
```

Bu satır, mysqli sınıfını kullanarak MySQL veritabanına bağlantı kurar. Burada "sunucu_adresi", "kullanıcı_adı", "parola", ve "veritabanı_adı" gerçek bilgilerinizle değiştirilmelidir.

```
if ($baglanti->connect_error) {
    die("Bağlantı hatası: " . $baglanti->connect_error);
}
```

Bu if bloğu, bağlantı sırasında bir hata olup olmadığını kontrol eder. Eğer bir hata varsa, hata mesajıyla birlikte script durdurulur.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

Veri tabanı bağlantısı örneklerle açıklanır.

$ad_mail = "SELECT adı, email FROM kullanıcılar";

Bu satırda, "kullanıcılar" tablosundan tüm kullanıcıların "adı" ve "email" bilgilerini seçecek SQL sorgusu $ad_mail değişkenine atanıyor.

$result = $baglanti->query($ad_mail);

Bu satır, $baglanti nesnesi üzerinden query() metodunu kullanarak daha önce hazırlanan $ad_mail SQL sorgusunu çalıştırır. Sorgu sonucu **$result** değişkenine atanır.

if ($result->num_rows > 0) {

Bu if bloğu, sorgu sonucunda dönen satır sayısının 0'dan büyük olup olmadığını kontrol eder. Yani, veritabanında aranan kriterlere uygun kayıt olup olmadığını kontrol eder.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

Veri tabanı bağlantısı örneklerle açıklanır.

```
while($row = $result->fetch_assoc()) {
        echo "Adı: " . $row["adı"]. " - Email: " . $row["email"]. "<br>";
    }
```

Bu döngü, sorgu sonucundaki her bir satırı $row değişkenine alır ve bu satırları ekrana yazdırır. fetch_assoc() metodu, sonuçları ilişkisel dizi (associative array) olarak döndürür, böylece sütun adlarına göre verilere erişilebilir.

```
} else {
echo "0 sonuç bulundu";
}
```

Bu else bloğu, sorgu sonucunda hiçbir satır dönmediyse (yani tabloda aranan kriterlere uygun kayıt bulunamadıysa) çalışır ve ekrana "0 sonuç bulundu" mesajını yazdırır.

```
$baglanti->close();
```

Bu satır, işlemler bittikten sonra veritabanı bağlantısını kapatır. Bu, kaynakların etkin bir şekilde yönetilmesi için önemlidir.

4.1. VERİ TABANI BAĞLANTISI YAPAR.

Hata ayıklama yolları gösterilir.

Bağlantı Hatası: Bağlantı hatası yaşadığınızda, PHP'nin hata mesajlarını göstermesini sağlayarak sorunun ne olduğunu anlayabilirsiniz.

```
$baglanti = new mysqli("sunucu_adresi", "kullanıcı_adı", "parola", "veritabanı_adı");

if ($baglanti->connect_error) {

    die("Bağlantı hatası: " . $baglanti->connect_error);

}
```

SQL Sorgu Hatalarını Kontrol Etme: Veritabanı bağlantısı başarılı olsa bile, SQL sorguları sırasında hatalar meydana gelebilir.

```
$sql = "SELECT * FROM tablo_adi";

if (!$result = $baglanti->query($sql)) {

    echo "Sorgu hatası: " . $baglanti->error;

}
```

Geliştirme Ortamında Hata Gösterimini Aktif Etme: Geliştirme aşamasındayken, PHP'nin tüm hata mesajlarını göstermesini sağlamak faydalı olabilir. php.ini dosyanızda veya PHP dosyanızın başında şu ayarları yaparak hata gösterimini aktif edebilirsiniz:

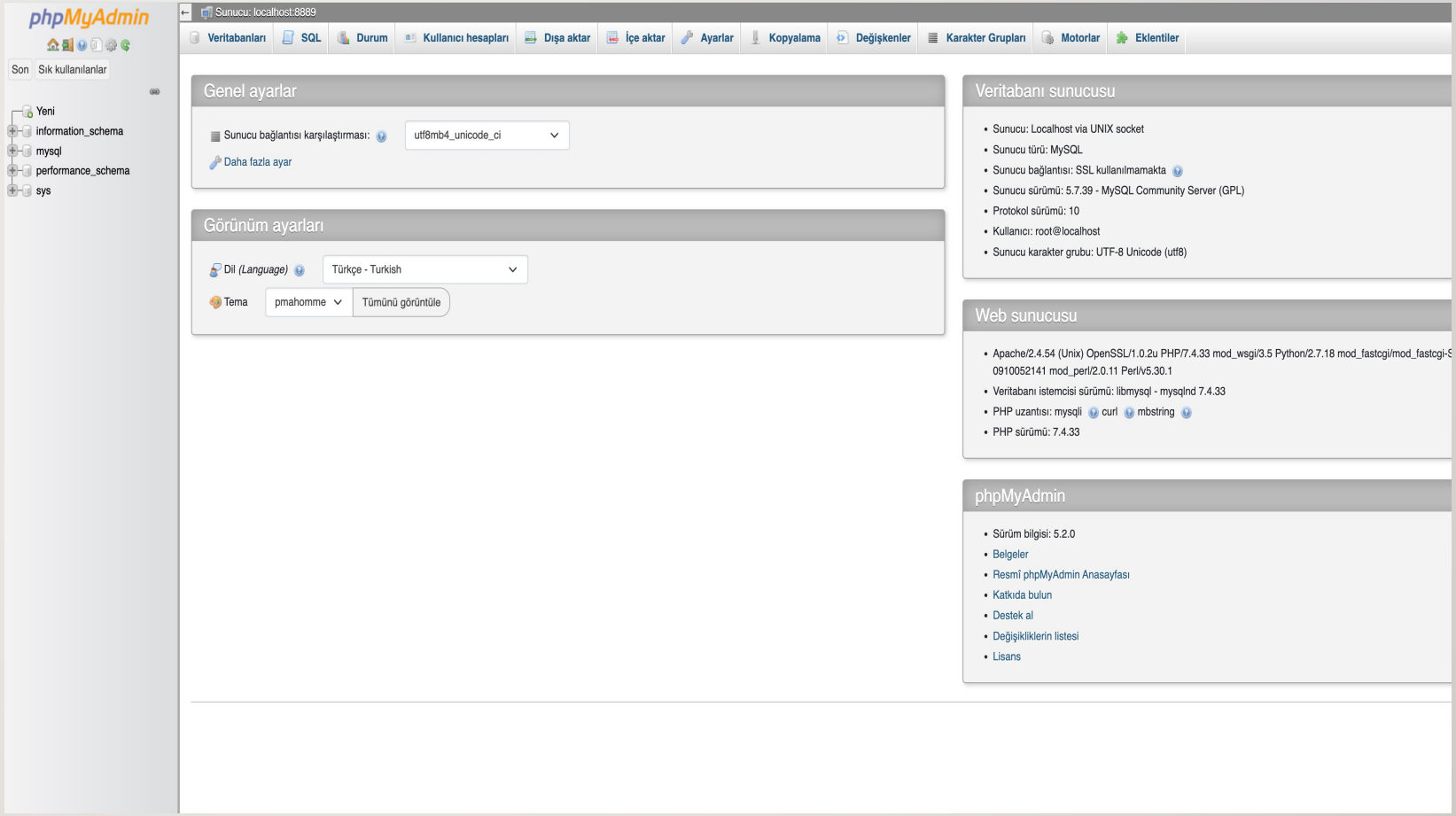
```
ini_set('display_errors', 1);

error_reporting(E_ALL);
```

4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

http://localhost:8887/phpMyAdmin5/

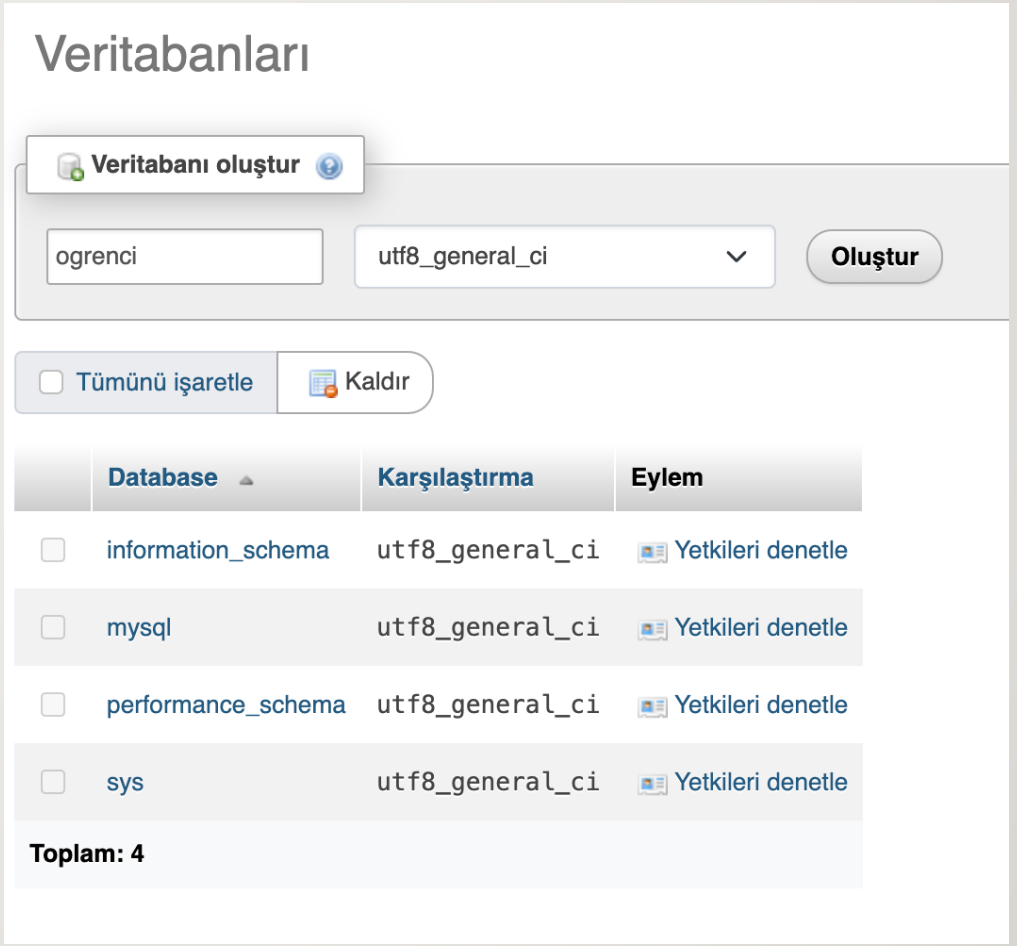
Phpmyadmin paneli



4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

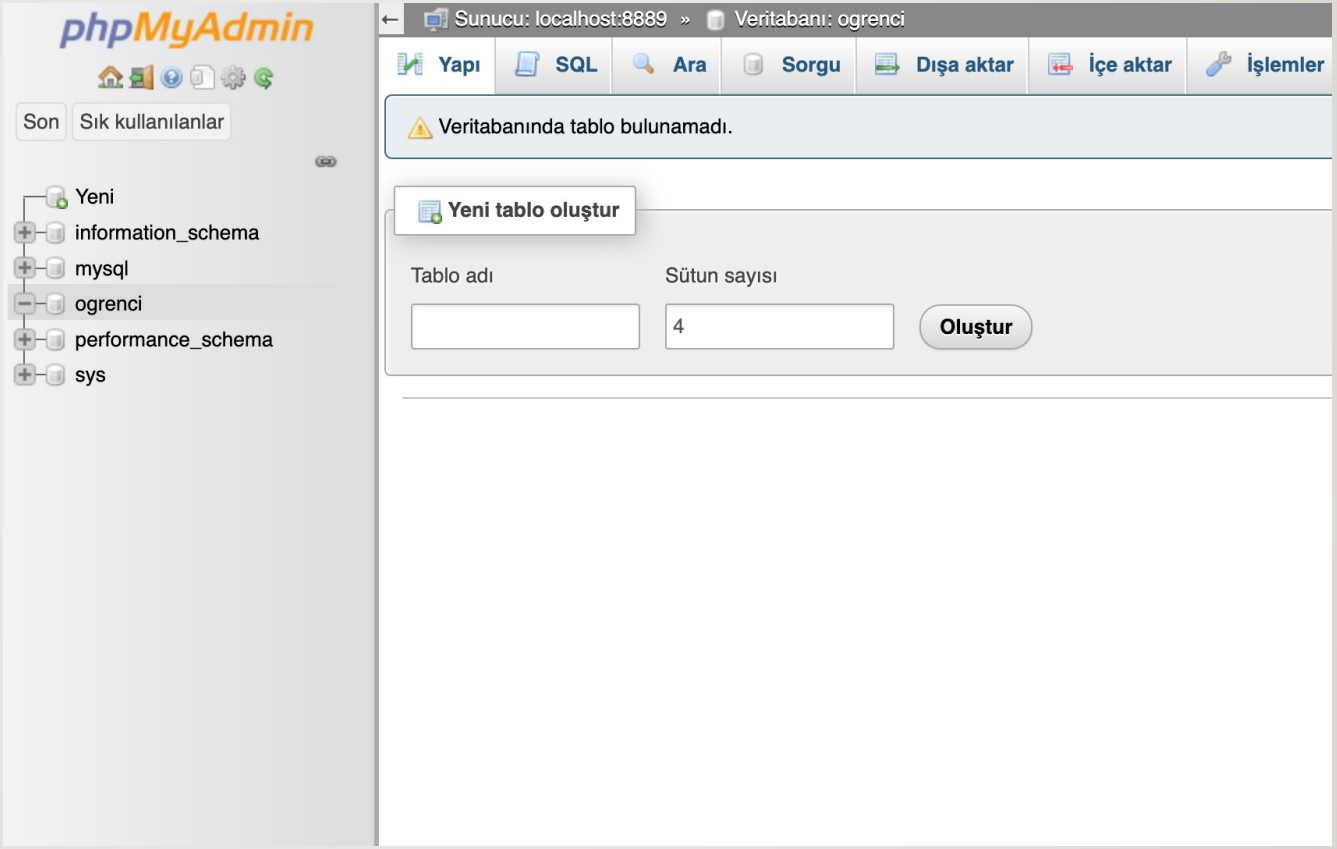
MYSQL VERİTABANI OLUŞTURMA



4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

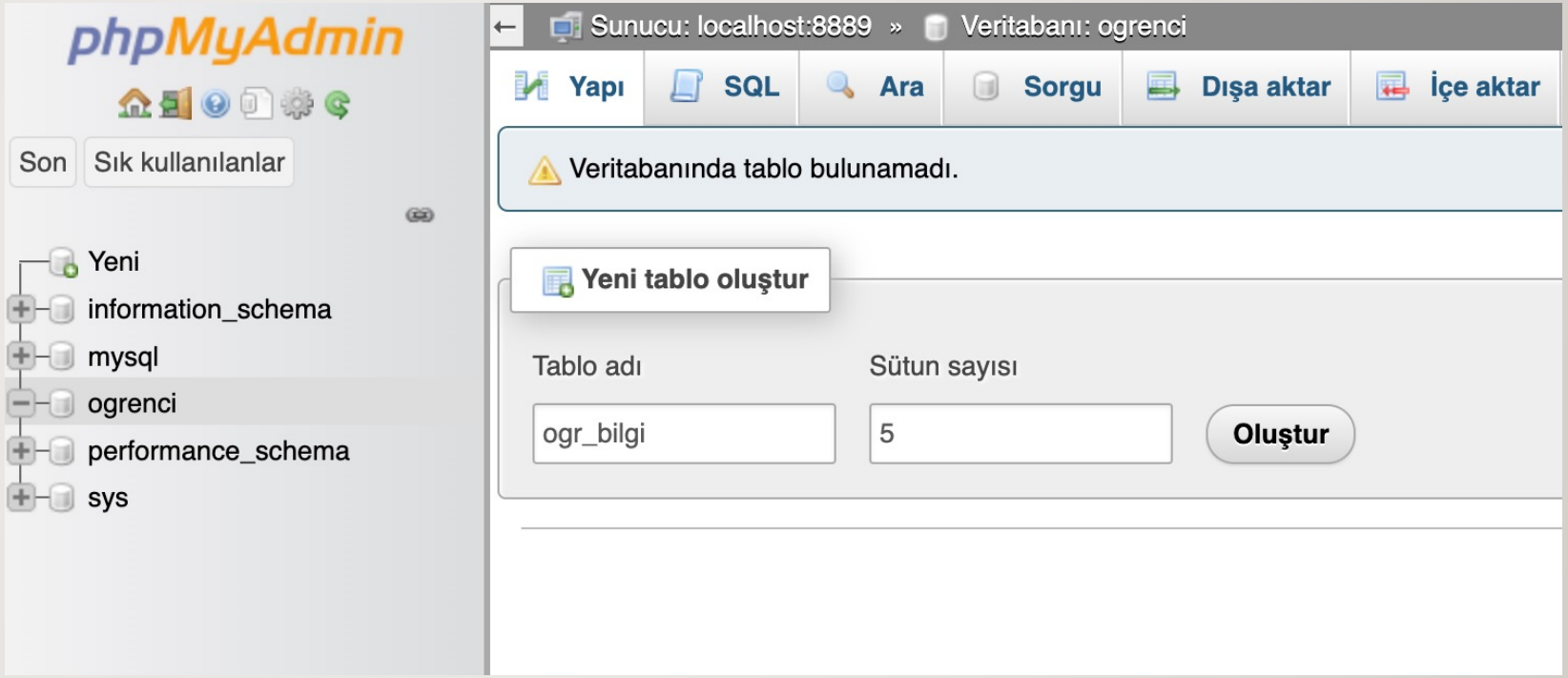
MYSQL TABLO OLUŞTURMA



4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

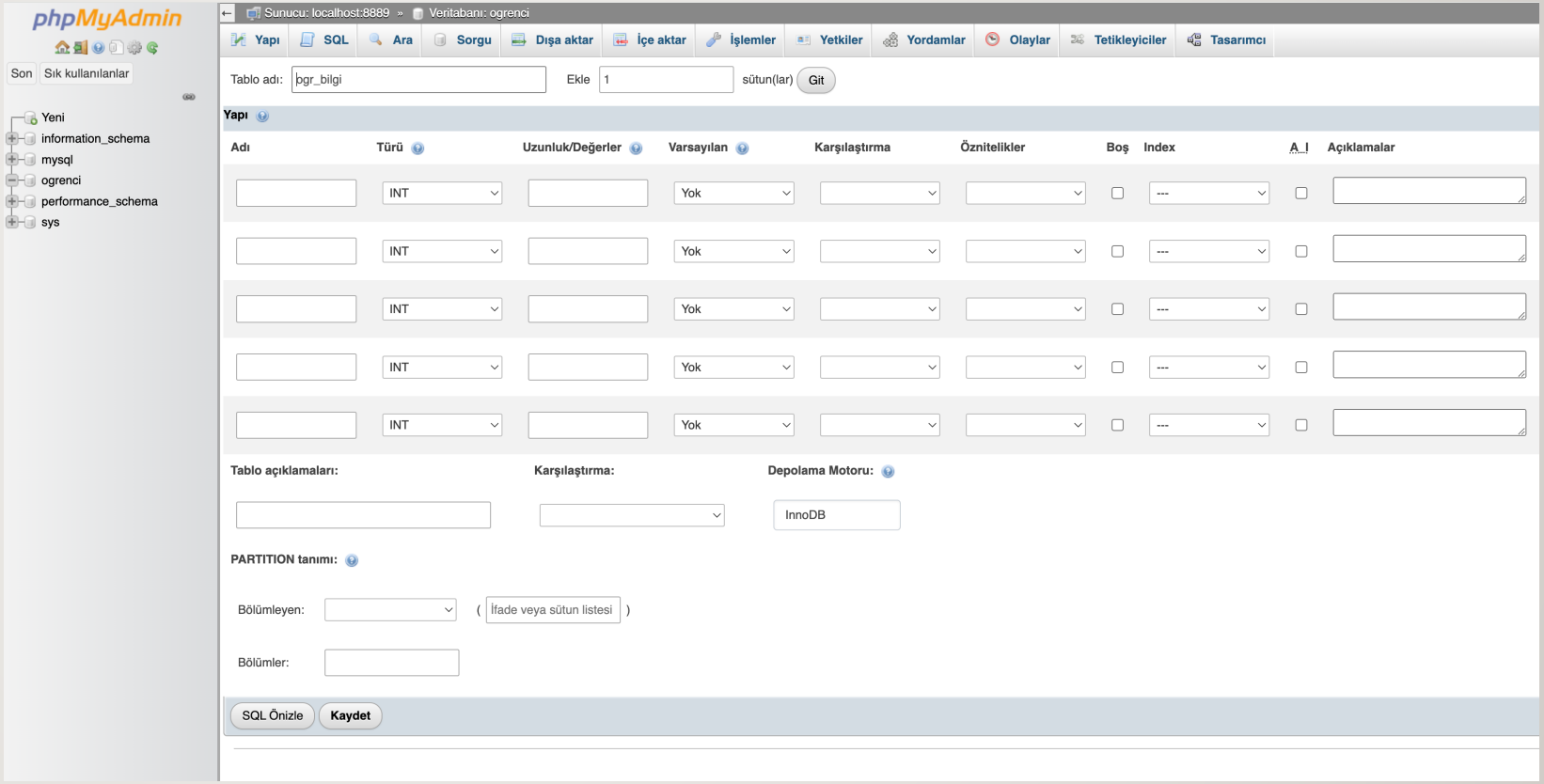
MYSQL TABLO OLUŞTURMA



4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

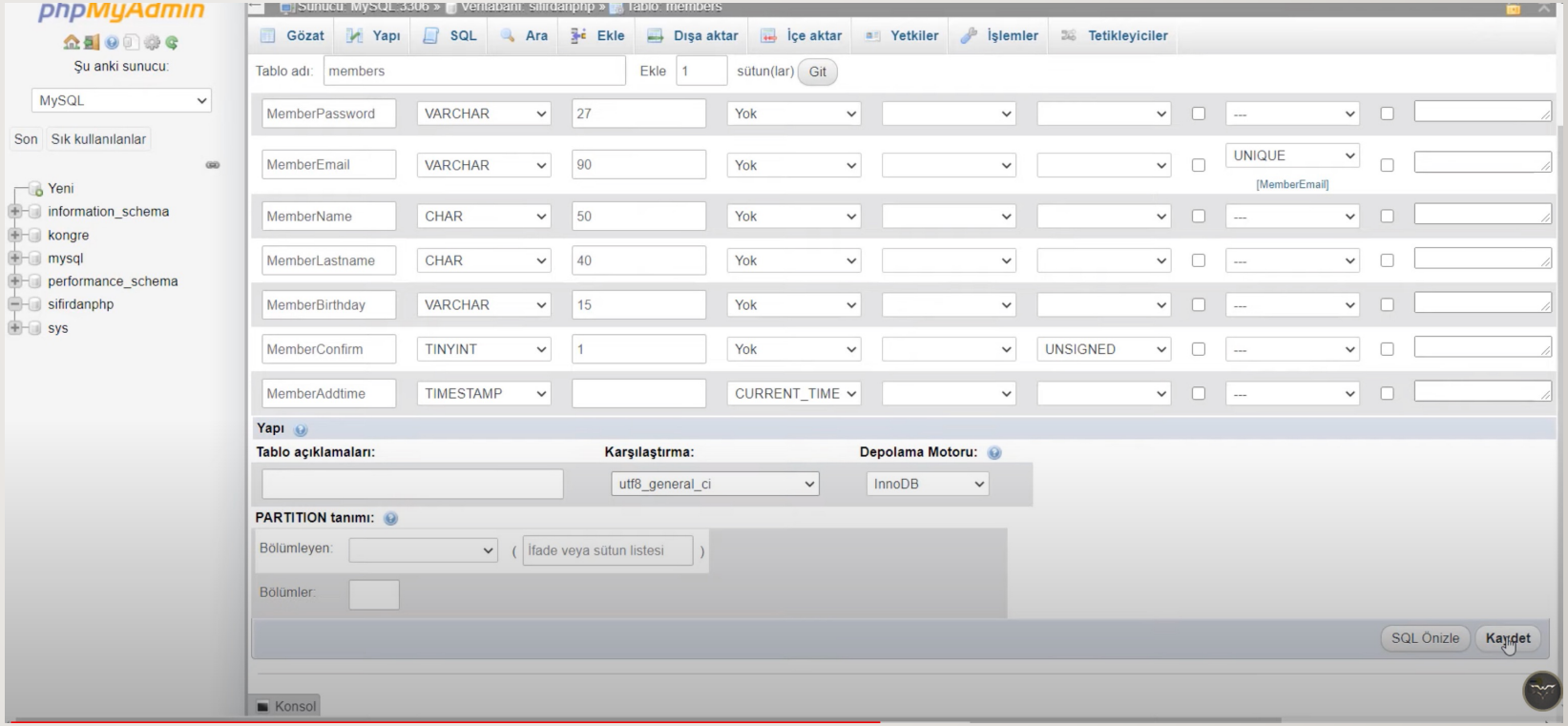
MYSQL TABLO OLUŞTURMA



4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

MYSQL TABLO OLUŞTURMA



4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

MYSQL VERİ TÜRLERİ

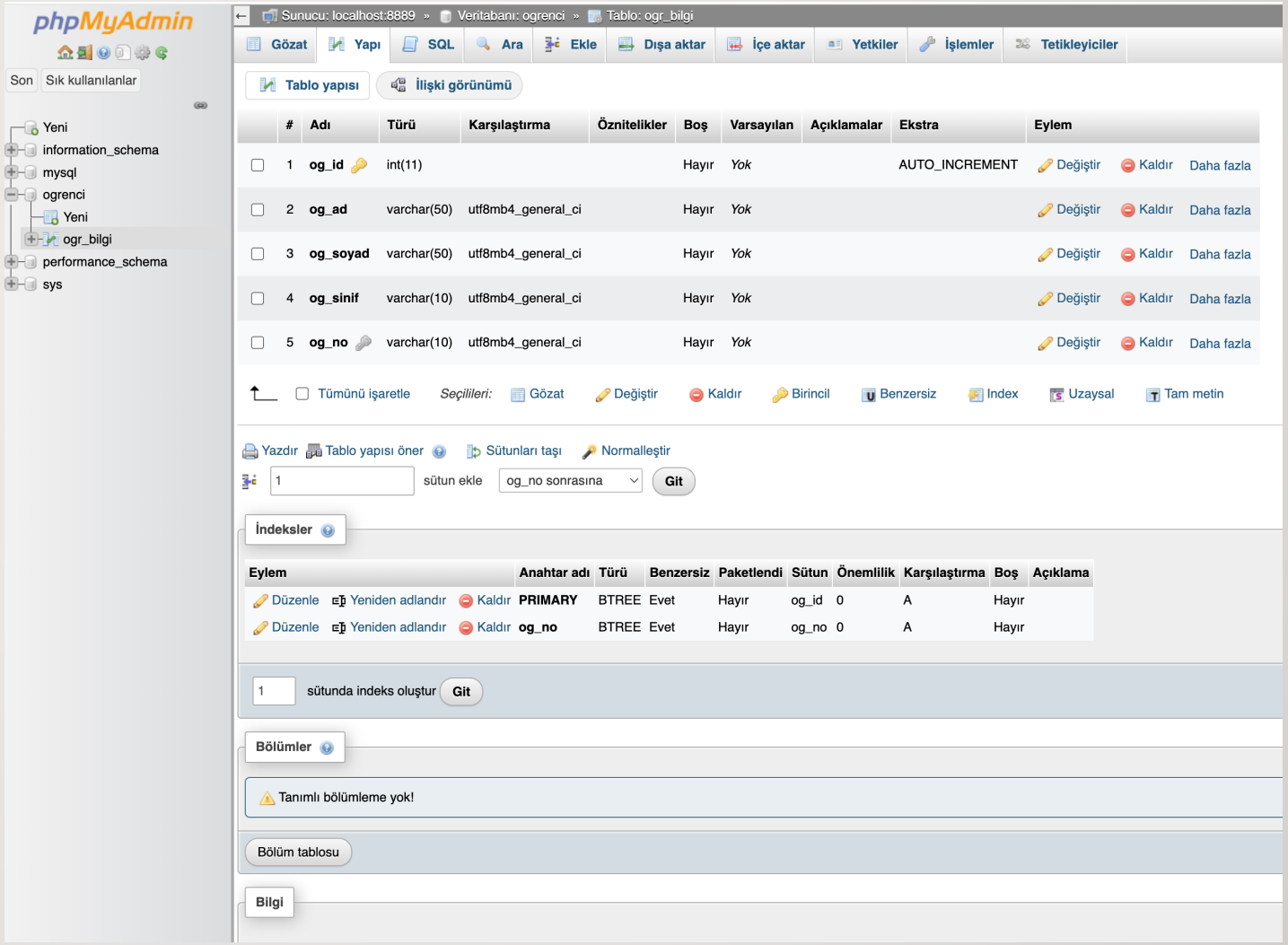
MySQL DATA TYPES

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

4.2. VERİ TABANI OLUŞTURUR.

PHP programlama dili komutları kullanarak veri tabanı, tablolar ve alanlar oluşturulur. Öğrencilerin veri tabanı, tablo ve alanlar oluşturmaları için gerekli kod satırlarını yazmaları istenir. Daha sonra yazdıkları kod satırları uygulamalı olarak çalıştırılarak sonuçları değerlendirilir.

MYSQL TABLO OLUŞTURMA



4.2. VERİ TABANI OLUŞTURUR.

PHP DİLİ İLE VERİ TABANI İŞLEMLERİ

```
<?php
// Veritabanı bağlantı bilgileri
$servername = "localhost";
$username = "username"; // Kullanıcı adınızı girin
$password = "password"; // Şifrenizi girin
$dbname = "ogrenci"; // Oluşturmak istediğiniz veritabanı adını girin

// MySQLi bağlantısını oluşturma
$conn = new mysqli($servername, $username, $password);

// Bağlantıyı kontrol etme
if ($conn->connect_error) {
    die("Bağlantı hatası: " . $conn->connect_error);
}
```

4.2. VERİ TABANI OLUŞTURUR.

PHP DİLİ İLE VERİ TABANI İŞLEMLERİ

```
// Veritabanı oluşturma SQL komutu
$sql = "CREATE DATABASE IF NOT EXISTS $dbname";
if ($conn->query($sql) === TRUE) {
    echo "Veritabanı başarıyla oluşturuldu\n";
} else {
    echo "Veritabanı oluşturulurken hata: " . $conn->error;
}

// Veritabanını seçme
$conn->select_db($dbname);
```

4.3. VERİ TABANI ÜZERİNDE İŞLEMLER YAPAR.

PHP DİLİ İLE VERİ TABANI İŞLEMLERİ

```
// INSERT (Oluşturma) işlemi
$sql = "INSERT INTO ogr_bilgi (og_ad, og_soyad, og_sinif, og_no) VALUES
('Ali', 'Yılmaz', ‘10A', '1001'),
('Ayşe', 'Kaya', ‘11B', '1002'),
('Mehmet', 'Çelik', ‘10B', '1003'),
('Elif', 'Demir', ‘12A', '1004'),
('Ahmet', 'Arslan', ‘11B', '1005'),
('Zeynep', 'Bulut', ‘10A', '1006')";
if ($conn->query($sql) === TRUE) {
    echo "Yeni kayıtlar başarıyla eklendi\n";
} else {
    echo "Kayıt eklerken hata: " . $conn->error;
}
```

4.3. VERİ TABANI ÜZERİNDE İŞLEMLER YAPAR.

PHP DİLİ İLE VERİ TABANI İŞLEMLERİ

```
// UPDATE (Güncelleme) işlemi
$sql = "UPDATE ogr_bilgi SET og_sinif = ‘11A' WHERE og_id = 1";
if ($conn->query($sql) === TRUE) {
    echo "Kayıt başarıyla güncellendi\n";
} else {
    echo "Güncelleme hatası: " . $conn->error;
}

// DELETE (Silme) işlemi
$sql = "DELETE FROM ogr_bilgi WHERE og_id = 6";
if ($conn->query($sql) === TRUE) {
    echo "Kayıt başarıyla silindi\n";
} else {
    echo "Silme hatası: " . $conn->error;
}
```

4.3. VERİ TABANI ÜZERİNDE İŞLEMLER YAPAR.

PHP DİLİ İLE VERİ TABANI İŞLEMLERİ

```
// SELECT (Okuma) işlemi
$sql = "SELECT * FROM ogr_bilgi";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    // Her satırı döngü ile işle
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["og_id"]. " - Ad: " . $row["og_ad"]. " " . $row["og_soyad"]. "
- Sınıf: " . $row["og_sinif"]. " - Numara: " . $row["og_no"]. "\n";
    }
} else {
    echo "Sonuç bulunamadı";
}

// Bağlantıyı kapama
$conn->close();
?>
```