Burdur
Mehmet Akif Ersoy
Üniversitesi

MAKÜ|GMYO
GÖLHİSAR MESLEK YÜKSEKOKULU

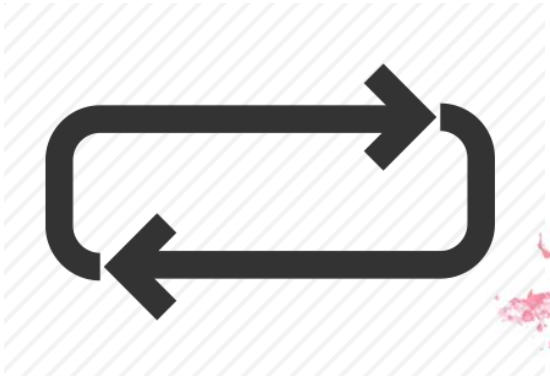
İnternet Programcılığı I

Öğr. Gör. Hüseyin Şengün
hsengun@mehmetakif.edu.tr

5. Hafta
Döngüler, Fonksiyonlar

Döngüler

- Programlama dillerinin en faydalı olduğu alanlardan biri de tekrarlanan işleri otomatikleştirmektedir.
- Eğer bir şeyi birkaç kez aynı ya da benzer şekilde yapmanız gerekiyorsa kodlarınızın bazı kısımlarını tekrar etmek için döngülere başvurabilirsiniz.
- *PHP'de şu döngü türleri yer almaktadır:*
- **while**
- **do-while**
- **for**
- **foreach**



while

- ***while döngüsü*** döngü tiplerinin en basit olanıdır. Bir *if* ifadesi gibi bu da bir koşula bağlıdır.
- Bir while döngüsü ile bir *if* ifadesi arasındaki fark, *if* ifadesinin eğer koşulu TRUE ise takip eden kod bloğunun bir kez çalışmasıdır.
- Bir while döngüsü ise *koşul TRUE olduğu sürece kod bloğunu çalıştırmayı sürdürür.*
- Bazen, while ifadesi daha başlangıçta FALSE değerini verir, bu durumda while etki alanındaki deyimler tek bir defa bile çalıştırılmazlar.

while

- Bir while deyiminin basit kullanımı:
- *while (Koşul)*
  {
  Kodlar;
  }

Not: if deyiminde olduğu gibi, birden çok deyimi aynı while döngüsü içinde süslü ***{...}*** parantezler arasında gruplayabilirsiniz.

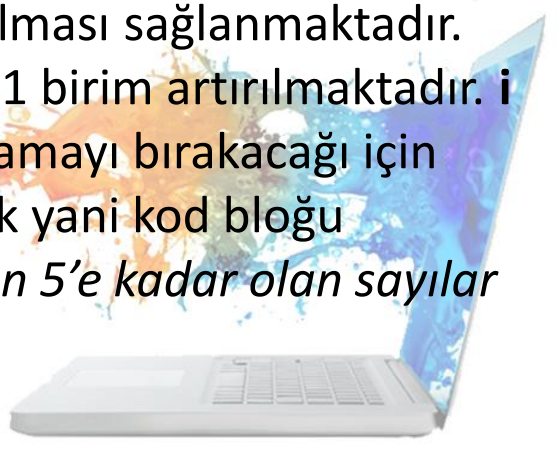
while

Örnek:

```php
<?php

    $i = 1;

    while($i <= 5) {
        echo "$i <br>";
        $i++;
    }

?>
```

```
1
2
3
4
5
```

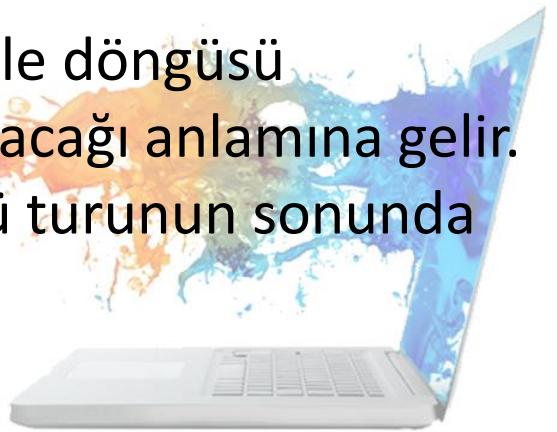
Bu örnekte **i** değişkenine 1 değeri atanmakta ve **while döngüsü** içerisinde 5'ten küçük ya da 5'e eşit olduğu sürece kod bloğunun çalıştırılması sağlanmaktadır. Döngünün her bir adımında **i** değişkeni ekrana yazdırılıp 1 birim artırılmaktadır. **i** değişkeni bir birim artış sonrası 6 olduğunda koşulu sağlamayı bırakacağı için ekrana yazdırma ve değerini artıma işlemleri sona erecek yani kod bloğu çalıştırılmayı bırakacaktır. *Tüm bu döngü aracılığı ile 1'den 5'e kadar olan sayılar ekrana alt alta yazdırılmış olacaktır.*



do-while

- ***do-while döngüsü***, while döngüsüne çok benzerdir. do-while döngüsünde farklı olarak, koşul ifadesi her döngünün başında değil sonunda test edilmektedir.
- ***Do***
  {
  kodlar;
  }
  while (Koşul)

  Koşul ifadesinin sonda yer alması da bir do-while döngüsü içerisindeki kod bloğunun en az bir kez çalıştırılacağı anlamına gelir. Çünkü kontrol aşaması, kod bloğu birinci döngü turunun sonunda devreye girer.



do-while

Örnek:

```php
<?php

    $i = 0;

    do {
        echo "$i <br>";
        $i++;
    } while ($i > 0);

?>
```

Yukarıdaki döngü tam olarak bir defa çalışacaktır (ekran çıktısı 0), ilk tekrardan sonra ifadenin doğruluğuna bakıldığında FALSE değerini verecek (***$i*** sıfırdan büyük değildir) ve döngünün çalışması sonlanacaktır.

do-while

Örnek:

```php
<?php

    $i = 1;

    do {
        echo "$i <br>";
        $i++;
    } while ($i <= 5);

). ?>
```

```
1
2
3
4
5
```

Yukarıdaki döngü while anlatımındaki örnek kod yapısının do-while döngüsüne dönüştürülmüş hali olup aynı sonuçları ekrana yazdıracaktır.

for

- ***for döngüsü;*** bir başlangıç değerine, kontrol sayacına ve artış miktarına sahip bir döngü türüdür.
- Temelde while ile benzer amacı taşımaktadır. Ancak while döngüsünde sınır değerler belirtilmek zorunda değilken for döngüsünde *başlangıç ve bitiş değerleri* baştan belirtilmelidir.
- **for** *(başlangıç değeri; kontrol sayacı; artış miktarı)*

```
{
        kodlar;
}
```

for

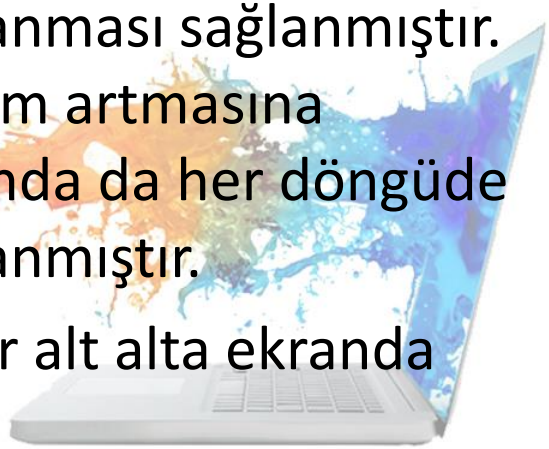
- ***başlangıç değeri:*** İlk başta, bir kere çalıştırılır. Burada genelde bir sayacın ilk değeri ayarlanır.
- ***kontrol sayacı(bitiş değeri):*** Bu ifade her döngüden önce kontrol edilir. Eğer ifade FALSE sonucunu döndürürse döngü durur, ifade TRUE olduğu sürece döngü devam eder. Burada genellikle bir limit belirtilir.
- ***artış miktarı:*** Her döngünün sonunda çalışır. Burada başlangıçta belirtilen sayacın değeri ayarlanır.
- ***Deyim(kodlar):*** Her döngüde bir kez çalıştırılır. Bu deyim genellikle bir kod bloğudur ve döngü ile asıl yapılmak istenen kodları barındırır.

for

- ***Örnek:***

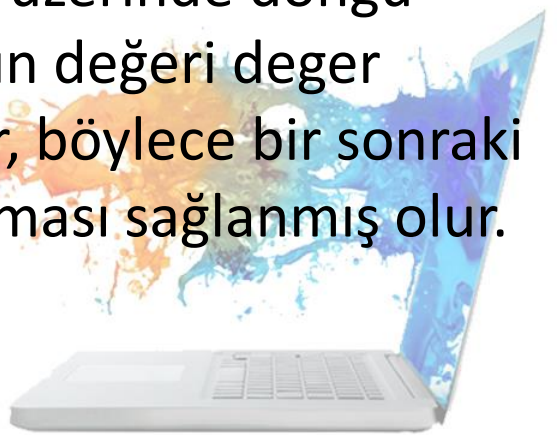
```php
<?php

    for ($y = 1; $y <= 10; $y++) {
        echo "$y <br>";
    }

?>
```

- Yukarıdaki örnekte başlangıç değeri olarak **y** değişkenine 1 değeri atanmıştır. Kontrol sayacı kısmında **y** değişkeninin 10'dan küçük veya 10'a eşit olduğu sürece döngünün tekrarlanması sağlanmıştır. Artış miktarı ise her döngü sonunda **y**'nin 1 birim artmasına ayarlanmıştır. Döngünün deyim yani kod bloğunda da her döngüde **y** değişkenin ekrana alt alta yazdırılması amaçlanmıştır.
- Bu döngü sonunda 1'den 10'a kadar olan sayılar alt alta ekranda görüntülenecektir.



foreach

- ***foreach döngüsü*** dizilerde kullanılan bir döngü türüdür. Bir dizinin elemanlarına bir döngü aracılığı ile tek tek ulaşılmasına olanak sağlar.
- **foreach** (dizi_ifadesi **as** $deger)
  {
  kodlar;
  }
- Bu sözdizimi ile dizi_ifadesi ile belirtilen dizinin üzerinde döngü oluşturur. Her yinelemede, sırası gelen elemanın değeri deger değişkenine atanır ve dizi göstericisi bir arttırılır, böylece bir sonraki yinelemede dizinin bir sonraki elemanına bakılması sağlanmış olur.



foreach

```php
<?php

    $arabalar = array("BMW", "Mercedes", "Ferrari", "Tesla");

    foreach ($arabalar as $deger) {
        echo "$deger <br>";
    }

?>
```

- Yukarıdaki örnekte **arabalar** adındaki bir dizi değişkeninde yer alan elemanlar foreach döngüsü aracılığıyla her bir yinelemede sırayla **deger** değişkenine atanmakta ve alt alta ekrana yazdırılmaktadır.

Ödev

- ***Dilediğiniz tipte bir döngüyle 1 ile 100 arasındaki "5'e tam bölünebilen" sayıları alt alta yazdırınız.***

- ***19 sayısının asal sayı olup olmadığını ekrana yazdıran bir for döngüsü kurunuz.***

Fonksiyonlar

- **Fonksiyonlar,** bir programda tekrar tekrar kullanılabilecek ifade bloklarıdır. Bu bloklar içerisinde geçerli her tür PHP kodu kullanılabilir. Bir fonksiyon, çağrı yapan bir arabirim gerektiren, belirli bir işlemi gerçekleştiren ve sonra da isteğe bağlı olarak bir sonuç döndüren bir modüldür. Dolayısıyla bu modüller sayfa yüklenir yüklenmez değil çağrıldığında çalışır.
- PHP'de hâlihazırda kullanılan birçok tanımlı fonksiyon bulunmaktadır. Bunların yanı sıra siz de kendi fonksiyonlarınızı yazabilirsiniz. Daha önceden PHP'de tanımlanmış herhangi bir fonksiyonun yapamadığı, tamamen size özgü işlevlere sahip bir fonksiyon geliştirebilirsiniz.

Fonksiyonlar

- ***Fonksiyon Tanımlama:***
  Kullanıcı tanımlı bir fonksiyon bildirimi, **function** kelimesi ile başlar. Belirlen fonksiyon adı da bir parantez **()** açıp kapatılarak bitirilir. Bu parantezin içinin dolu olup olmaması fonksiyonun **parametreli** ya da **parametresiz** bir fonksiyon olduğunu, süslü parantezler {} ise fonksiyonun kapsam alanını gösterir.

Fonksiyonlar

```php
<?php

    function fonksiyonAdi() {
        // Çalıştırılacak kodlar buraya yazılacak
    }

?>
```

- Fonksiyon isimleri, PHP'deki diğer isimlerle aynı kurallara tabidir. Geçerli bir fonksiyon ismi bir harf ya da alt çizgi ile başlar, herhangi bir sayıda geçerli harf, sayı ya da alt çizgi ile devam eder. **Fonksiyon isimlerinde büyük-küçük harf duyarlılığı yoktur.** Yani mesajYaz(), mesajyaz() ya da MESAJYAZ() fonksiyonları birebir aynı fonksiyonlardır.

Fonksiyonlar

```php
<?php

    function mesajYaz() {
        echo "Merhaba dünya!";
    }

    mesajYaz(); // fonksiyonu çağırma

?>
```

- mesajYaz() fonksiyonu çağrıldığı her yere “Merhaba Dünya” çıktısını gönderecektir.

Fonksiyonlar

- ***Parametreli Fonksiyonlar***
- tekrar tekrar kullanılabilecek kod bloklarına bir ya da birden çok değişken aracılığıyla dinamik veri-veriler gönderme yöntemidir.

```php
<?php

    function fonksiyonAdi($deger) {
        echo "Merhaba $deger"; // $deger değişkeni dinamik olarak çıktıya eklenecek
    }

?>
```

- Bir fonksiyonu varsayılan bir parametre değeri atayarak da kullanabilirsiniz. Fonksiyona bir parametre gönderirseniz son gönderdiğiniz değer aktif olacak, parametresiz bir çağrı yaparsanız da varsayılan değer devreye girecektir.

Fonksiyonlar

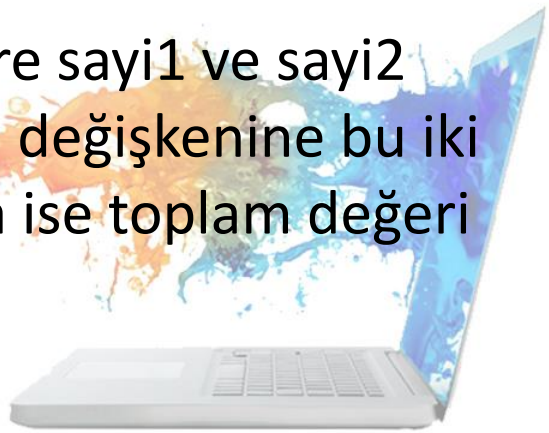
```php
<?php

    function yukseklikAyarla($yukseklik=40) {
        echo "Yükseklik: $yukseklik";
    }

    yukseklikAyarla(10); // Ekran çıktısı -> Yükseklik: 10
    yukseklikAyarla(); // Ekran çıktısı -> Yükseklik: 40
    yukseklikAyarla(100); // Ekran çıktısı -> Yükseklik: 100

?>
```

- Parametreli bir fonksiyonda birden çok değer de gönderebilirsiniz. Dikkat edilmesi gereken şey her bir değerin virgüllerle ayrılarak yazılmasıdır.

Fonksiyonlar

```php
<?php

    function topla($sayi1, $sayi2) {
        $toplam = $sayi1 + $sayi2;
        echo "Toplam: $toplam";
    }

    topla(8, 16); // Ekran çıktısı -> Toplam: 24

?>
```

- topla() fonksiyonu ile gönderilecek iki parametre sayi1 ve sayi2 değişkenleri ile yakalanacak daha sonra toplam değişkenine bu iki sayının toplamı atanacaktır. Fonksiyon sonunda ise toplam değeri ekrana yazdırılacaktır.



Fonksiyonlar

- ***Değer Döndürme:***
  Bazı durumlarda fonksiyonlar içerisinde bir **echo** komutu kullanarak ekran çıktısı almak istemeyebilirsiniz. Bu durumlarda fonksiyonda üretilen sonucu bir **return** ifadesi ile ekrana yazdırmadan döndürebilirsiniz. Dönen değer üzerinde dileğiniz işlemleri - ***ekrana yazdırma, değerini değiştirme, format dönüştürme -*** yapabilirsiniz.

Fonksiyonlar

- ***Değer Döndürme:***

```php
<?php

    function topla($sayi1, $sayi2) {
        $toplam = $sayi1 + $sayi2;
        return $toplam;
    }

    topla(8, 16); // Ekran çıktısı olmayacaktır

    echo topla(8, 16); // Ekran çıktısı -> 24

    echo "Sonuç: ".(topla(8, 16) + 3); // Ekran çıktısı -> Sonuç: 27

?>
```

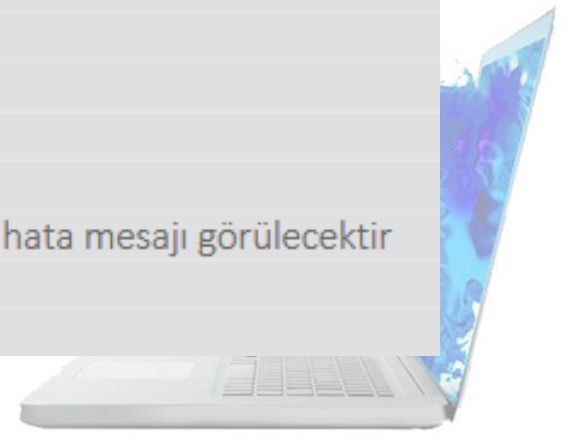
Fonksiyonlar

- **Fonksiyonlarda global ifadesi:**
- Fonksiyonlar varsayılan olarak kendi kapsamlarında tanımlanan değişkenleri ya da parametre ile gelen değişkenleri tanırlar. Bu kapsamlar dışında tanımlanan değişkenler ise tanınmazlar. Aynı zamanda kendi kapsamında tanımlanan değişkenleri de fonksiyon dışında kullandırmazlar.

Fonksiyonlar

- **Fonksiyonlarda global ifadesi:**

```php
<?php

    function isimOlustur() {
        $isim = 'Gazi';
    }

    isimOlustur();
    echo $isim; // isim adında bir değişkenin tanımlanmadığına yönelik hata mesajı görülecektir

    $deger = "Beltek";

    function degerYazdir() {
        echo $deger;
    }

    degerYazdir(); // deger adında bir değişkenin tanımlanmadığına yönelik hata mesajı görülecektir

?>
```



Fonksiyonlar

- Değişkenlerin tanımlanmadığına yönelik bu hataları almamak için **global** ifadesinin değişkenlerle birlikte bir kez kullanılması gerekir. Bu sayede herhangi bir yerde tanımlanan bir değişken hem fonksiyon içinde ve hem de fonksiyon dışında geçerlik kazanmış olacaktır.

Fonksiyonlar

```php
<?php

    function isimOlustur() {
        global $isim; // isim değişkeni fonksiyonun dışında da kullanılabilir hâle geldi
        $isim = 'Gazi';
    }

    isimOlustur();
    echo $isim; // Ekran çıktısı -> Gazi

    $deger = "Beltek";

    function degerYazdir() {
        global $deger; // deger değişkeni fonksiyonun içinde de kullanılabilir hâle geldi
        echo $deger;
    }

    degerYazdir(); // Ekran çıktısı -> Beltek

?>
```

Kaynaklar

- PHP Kılavuzu - http://php.net/manual/tr/index.php
- PHP Tutorials - https://www.w3schools.com/php/default.asp
- PHP Videolu Temel Dersler İçin: Youtube/Php Türkiye
- https://php-mysql.org/