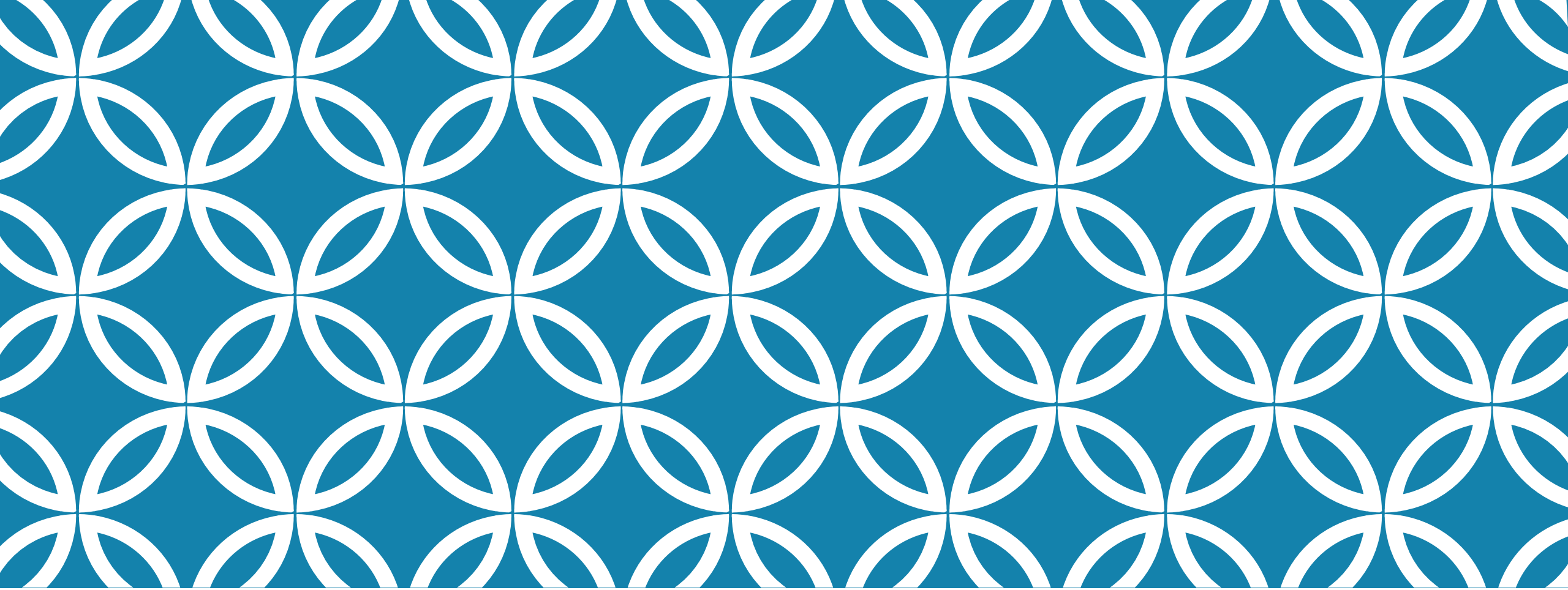

PROGRAMLAMA TEMELLERİ

Öğr. Gör. Erhan AKAGÜNDÜZ

FONKSİYONLAR

- Bir bilgisayar programı yazılırken bazı işlemlerin programın farklı yerlerinde sürekli tekrarlanması gerekebilir.
- **Örneğin;** arazi hesapları ile ilgili bir program yazılıyorsa sık sık geometrik şekillerin alanı hesaplanmak zorunda kalınabilir.
- Her gerektiğinde alan hesabı işlemini yerine getiren kodları yazmak hem programcının iş yükünü hem de hata yapma olasılığını arttırır.
- Bu nedenle programcılar, sık tekrarlanan işler için aynı kodu defalarca yazmak yerine, işlemi yerine getiren kod bloğunu yazıp adlandırarak ihtiyaç hâlinde bu adla basit bir şekilde çağırıp kullanmayı tercih ederler.

FONKSİYONLAR

- İhtiyaç duyulduğunda çağrılıp çalıştırılabilen bu kod paketlerine **fonksiyon** adı verilir.
- Fonksiyon farklı programlama dillerinde **prosedür** veya **yordam** olarak da adlandırılabilir.
- Sık tekrarlanan işlemleri gerektiğinde kullanılacak küçük kod parçalarına bölüp yazma yani "fonksiyon" yaklaşımı bütün programlama dillerinde kullanılabilecek bir yöntemdir.

FONKSİYONLAR

Fonksiyonlar sayesinde;

- Programcı aynı kodları defalarca yazma yükünden kurtulur.
- Daha az kod yazılacağı için hata yapma olasılığı azalır.
- Fonksiyon sadece çağrıldığında kullanılacağı için bilgisayarın bellek kullanımından tasarruf edilir.
- Kod okunabilirliğini arttırır ve kod analizini daha kolay hâle getirir.
- Karmaşık problemlerin daha basit küçük parçalara ayrılarak çözülmesini kolaylaştırır.

FONKSİYONLAR

- Fonksiyonlar çağrıldıklarında, barındırdıkları kod kümelerini işleyerek oluşan sonuçları döndürebilir.
- Ayrıca istenirse kendilerine parametre olarak gönderilen verileri işleyip ürettikleri sonucu da döndürebilir.

FONKSİYONLARIN KULLANIMI

- Bu bölüme kadar yazılan örnek programlarda programlama dilinin bazı hazır fonksiyonları kullanıldı.
- Örneğin ekrana veri yazdırmak için kullandığınız **print()** bir fonksiyondur.
- Programlama dilleri yazılımcının gerektiğinde kullanabileceği birçok hazır fonksiyonla beraber gelir.
- Bunlara **built-in** yani **gömülü fonksiyonlar** denir.

GÖMÜLÜ FONKSİYONLARIN VE MODÜLLERİN KULLANIMI

- Programlama dili ile temel işlemleri yerine getiren birçok fonksiyon hazır ve tanımlanmış olarak gelir.
- Şu ana kadar kullanılmış olan **print**, **input**, **type**, **int**, **float**, **str** gibi fonksiyonlar programlama dili içinde gömülüdür.
- Gömülü fonksiyonlar, geliştiricileri tarafından programlama dili içine gömülmüş ve tanımlamaya gerek kalmadan kullanılabilen fonksiyonlardır.
- Gömülü fonksiyonlarda tek yapılması gereken fonksiyonu çağırmak ve kullanmaktır.

GÖMÜLÜ FONKSİYONLARIN VE MODÜLLERİN KULLANIMI

- Bu gömülü fonksiyonlar haricinde farklı işlevler için geliştirilmiş fonksiyon kütüphaneleri de vardır.
- Örneğin matematik işlemlerinde ihtiyaç duyabileceğiniz tüm fonksiyonlar, hazır olarak programlama dili ve “**Math**” isimli bir kütüphane ile gelir.
- İhtiyaç duyduğunuzda makine öğrenmesi, oyun geliştirme, ağ işlemleri gibi alanlarda size gerekli işlevleri sağlayacak kütüphaneler programlama diline eklenip kullanılabilir.

GÖMÜLÜ FONKSİYONLARIN VE MODÜLLERİN KULLANIMI

- Kendiniz de projenizde kullanmak için yazdığınız fonksiyonları bir kütüphane hâlinde toplayarak ihtiyacı olan programcılara dağıtabilirsiniz.
- Bu şekilde bir konuda belirli işlevleri yerine getiren fonksiyonların bir araya getirildiği Python dosyalarına **modül** denir.

GÖMÜLÜ FONKSİYONLARIN VE MODÜLLERİN KULLANIMI

- Programlama diline eklenmiş olan modülü ve içerdiği fonksiyonları kullanabilmek için önce yazılan kodun başına “import” komutu eklenerek modüle erişim sağlanır.
- Programlama dili kurulumu ile gelen, matematik fonksiyonlarını içeren “math.py” dosyasına yani **Math modülüne** erişmek için programın başına aşağıdaki gibi erişim ifadesi eklenmesi gerekir.
- **from** modül_adi **import** fonksiyon_adi

GÖMÜLÜ FONKSİYONLARIN VE MODÜLLERİN KULLANIMI

- Bu satırla programa modülden karekök bulan **sgrt()**, güç hesabı yapan **pow()** ve trigonometri işlemi yapan **sin()** ve **cos()** fonksiyonlarına erişim imkânı verilmiş olur.

```
from math import sin, sqrt, cos, pow # fonksiyonlara erişim sağlıyoruz.
print( sqrt(4) ) # 4 sayısının karekökünü buldurup ekrana yazdırıyoruz.
print( sin(30) ) # 30 sayısının sinüs değeri
print( cos(45) ) # 45 sayısının cosinüs değeri
print( pow(3,2) ) # 3'ün 2. kuvveti
```

```
2.0
-0.9880316240928618
0.5253219888177297
9.0
```

GÖMÜLÜ FONKSİYONLARIN VE MODÜLLERİN KULLANIMI

- Aşağıdaki örnek programda Math modülündeki bütün fonksiyonlar erişime açılmış, bazıları kullanılmıştır.

```
import math
print( math.pow(3,12) )
print( math.sqrt(9) )
print( math.sin(math.pi/2) )
```

```
531441.0
3.0
1.0
```

FONKSİYON TANIMLAMA

Fonksiyonlar **def** komutu kullanarak tanımlanabilir. Fonksiyon tanımlarken izlenecek yol aşağıdaki gibidir:

1. **def** komutu yazılarak yeni bir fonksiyon tanımlanacağı programlama diline bildirilir.
2. Anahtar sözcükten sonra fonksiyon çağrılırken kullanılacak olan isim Python'un isimlendirme kurallarına uygun olarak belirlenmelidir. Burada fonksiyonun işlevi ile ilintili bir isim vermek kod okunabilirliği açısından mantıklı olacaktır.

FONKSİYON TANIMLAMA

3. Parantezler arasına fonksiyona gönderilecek parametreler yazılır, eğer fonksiyonumuz parametre almıyorsa parantez araları boş bırakılır. Tanım sonuna iki nokta üst üste konarak alt satırdan itibaren kod bloğunun başladığı belirtilir.

4. Tanım ve isim satırının altında bir sekme (tab) boşluk bırakılarak fonksiyon çağrıldığında çalışacak kodlar yazılır.

```
def fonksiyon_adi  ( varsa parametre listesi ) :
  Kod_blogu
  Kod_blogu
```

FONKSİYON TANIMLAMA

- **Örnek**: Selamla isminde bir fonksiyon tanımlayıp 3 kez kullanınız.

```
def selamla():
    print("Merhaba Özalp MYO")

selamla() #fonksiyonu çağırıyoruz
selamla() #fonksiyonu 2. defa çağırıyoruz
selamla() #fonksiyonu 3. defa çağırıyoruz
```

```
Merhaba Özalp MYO
Merhaba Özalp MYO
Merhaba Özalp MYO
```

- **NOT:** Bir programda fonksiyon kullanılmadan önce tanımlanmalıdır, aksi hâlde programınız hata verecektir.

FONKSİYON TANIMLAMA

- **Örnek:** 1-10 arasındaki sayıları bir fonksiyon tanımlayarak yazınız.

```
def onakadar_say():
    for i in range(1,10):
        print("Bilgisayar Teknolojileri")
```

```
onakadar_say()
```

```
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
Bilgisayar Teknolojileri
```

```
def ondefa_yaz():
    a = 0
    while a < 10:
        a += 1
        print("Python Programlama")
```

```
ondefa_yaz()
```

```
Python Programlama
Python Programlama
Python Programlama
Python Programlama
Python Programlama
Python Programlama
Python Programlama
Python Programlama
Python Programlama
Python Programlama
```

PARAMETRE KAVRAMI VE FONKSİYONLAR İLE PARAMETRE KULLANIMI

- Şu ana kadar fonksiyon tanımlarken parantez içleri hep boş bırakıldı.
- Bu şekilde parametre kullanmayan sadece içerdiği kod bloğunu işleyen fonksiyonlar yazıldı.
- Fonksiyonların en önemli işlevlerinden biri de verileri parametre olarak alıp işledikten sonra size sonucu bildirebilme yetenekleridir.
- Parametre, yazılan fonksiyona işlemesi için gönderilen veridir.

PARAMETRE KAVRAMI VE FONKSİYONLAR İLE PARAMETRE KULLANIMI

- Metin, sayı, liste ve benzeri veriler fonksiyonlara işlenmeleri için parametre (bazen **referans** da denir) olarak gönderilebilir.
- Bunun için fonksiyon tanımlanırken parantez içerisine gönderilecek parametrenin hangi adla işleneceğinin belirtilmesi yeterlidir.

```
def selamla(ad):
    print("Merhaba", ad)

selamla("Ahmet")
selamla("Mehmet")
selamla("Samet")
```

```
Merhaba Ahmet
Merhaba Mehmet
Merhaba Samet
```

PARAMETRE KAVRAMI VE FONKSİYONLAR İLE PARAMETRE KULLANIMI

- Fonksiyon yazılırken alacağı parametrenin boş geçilmesi durumunda, parametre olarak varsayılan bir değer alacak şekilde düzenlenebilir.

```
def ulke_yaz(ulke = "Türkiye"): # varsayılan ülke değeri "Türkiye"
    print( ulke + " benim memleketim.")

ulke_yaz ("Türkiye")
ulke_yaz ("Azerbaycan")
ulke_yaz () #burada varsayılan değer çalışacak
ulke_yaz ("Almanya")
```

```
Türkiye benim memleketim.
Azerbaycan benim memleketim.
Türkiye benim memleketim.
Almanya benim memleketim.
```

PARAMETRE KAVRAMI VE FONKSİYONLAR İLE PARAMETRE KULLANIMI

- Parametre adlandırmaları biliniyorsa fonksiyon çağrılırken **doğrudan parametrelere atama** yapılabilir.

```
def en_kucuk_cocuk(cocuk3, cocuk2, cocuk1):
    print("En Genç olan cocuk " + cocuk3)

en_kucuk_cocuk (cocuk1= "Seyhan", cocuk2= "Ahmet", cocuk3= "Mehmet")
```

```
En Genç olan cocuk Mehmet
```

PARAMETRE KAVRAMI VE FONKSİYONLAR İLE PARAMETRE KULLANIMI

- Parametre adlandırmaları biliniyorsa fonksiyon çağrılırken **doğrudan parametrelere atama** yapılabilir.

```
def en_kucuk_cocuk(cocuk3, cocuk2, cocuk1):
    print("En Genç olan cocuk " + cocuk3)

en_kucuk_cocuk (cocuk1= "Seyhan", cocuk2= "Ahmet", cocuk3= "Mehmet")
```

```
En Genç olan cocuk Mehmet
```

PARAMETRE KAVRAMI VE FONKSİYONLAR İLE PARAMETRE KULLANIMI

- Fonksiyona parametre olarak nesne, dizi ya da koleksiyon da gönderilebilir.

```
def mevsim_yaz(mevsim_dizisi):
    for x in mevsim_dizisi:
        print(x)

mevsimler = ["İlkbahar", "Yaz", "Sonbahar", "Kış"]

mevsim_yaz(mevsimler)
```

```
İlkbahar
Yaz
Sonbahar
Kış
```