Python Programlama

Ders 4

Ali Mertcan KOSE Msc.
amertcankose@ticaret.edu.tr

İstanbul Ticaret Üniversitesi

Örnek Projektör

Projektör bir kara kutudur.

Nasıl çalıştığı bilinmiyor.

Arayüz giriş/çıkışı bilinir.

Bu girdi ile iletişim kurabilecek herhangi bir elektronik bağlayın.,

Kara Kutu bir şekilde görüntüyü giriş kaynağından bir duvara dönüştürür ve büyütür.

Soyutlama Fikri: projektörü kullanmak için nasıl çalıştığını bilmemize gerek yoktur.

Örnek Projektör

Olimpiyatlar için büyük bir görüntünün yansıtılması, ayrı projektörler için ayrı görevlere ayrılır.

Her projektör girdi alır ve ayrı çıktı üretir.

Tüm projektörler daha büyük görüntü üretmek için birlikte çalışır.

Ayrıştırma fikri: farklı cihazlar bir nihai hedefe ulaşmak için birlikte çalışır.

Ayrıştırma ile Yapı Oluşturun

Projektör örneğinde, ayrı cihazlar ayrışmaya izin verir.

Programlamada, kodu aşağıdaki modüllere bölün:

- Kendini içeren
- kodu bölmek için kullanılır.
- Yeniden kullanılabilir olması amaçlanmıştır.
- Kodu düzenli tutun.
- Kodu tutarlı tutun.

Şimdi fonksiyonlarla ayrışmanın nasıl elde edileceğini keşfedeceğiz. Daha sonra sınıflarla ayrışma sağlayacağız.

Fonksiyonlar

Bir isimle talimat dizisi.

Yeniden kullanılabilir parçalar/kod parçaları

Örnek olarak; round(), randint(), len()

Fonksiyonlar bir programda "çağrılana" veya "çağrılana" kadar çalıştırılmaz.

Fonksiyon özellikleri:

- Bir Adı Var.
- Parametreleri vardır (0 veya daha fazla)
- Bir belge dizisi var (isteğe bağlı ancak önerilir).
- Bir gövdesi vardır.
- Bir şey döndürür.

return İfadesi

`return` sadece bir fonksiyonun içinde anlamı vardır.

Bir işlev içinde yalnızca bir dönüş yürütülür.

İşlevin içindeki herhangi bir kod, ancak return ifadesinden sonra yürütülmelidir.

Kendisiyle ilişkilendirilmiş bir değeri var, işlev çağırana döndürülür.

İşlevleri Yazma ve Çağırma

```
def is_even(i): “ “ ” Input: i, a positive int
Returns True if i is even, otherwise False
“ “ ” print (“inside is_even”)
return i%2==0
is_even(3)
```

Fonksiyon Gövde

```
def is_even(i):
" " " Input: i, a positive int Returns True if i is even, otherwise False
" " " print ("inside is_even")
return i%2==0
```

Değişken Kapsamı

Bir değişkenin kapsamı, değişkene erişilebildiği programın bir parçasıdır.

Yerel değişkenler, bir işlev içinde tanımlanan değişkenlerdir.

Yerel değişkenler, bir blokta tanımlandıkları noktadan, tanımlandığı fonksiyonun sonuna kadar kullanılabilir hale gelir.

İşlevlerin dışında tanımlanan değişkenlerin genel bir kapsamı vardır. Bu, göreceğimiz gibi, değerlerine tüm işlevlerin içinden erişilebileceği, ancak güncellenemeyecekleri anlamına gelir.

Değişken Kapsamı ve Fonksiyonlar

Biçimsel parametrelere, işlev çağrısında iletilen değerlere, **gerçek parametrelere** atanacaktır.

Bir işlev girildiğinde yeni kapsam/çerçeve/ortam oluşturulur.

```
def f(x):
x=x+1
print ('in f(x):x=',x)
return x
x=3
f(x)
```

Değişken Kapsamı

```
def f( x ):
    x = x + 1
    print('in f(x): x =', x)
    return x

x = 3
z = f( x )
```

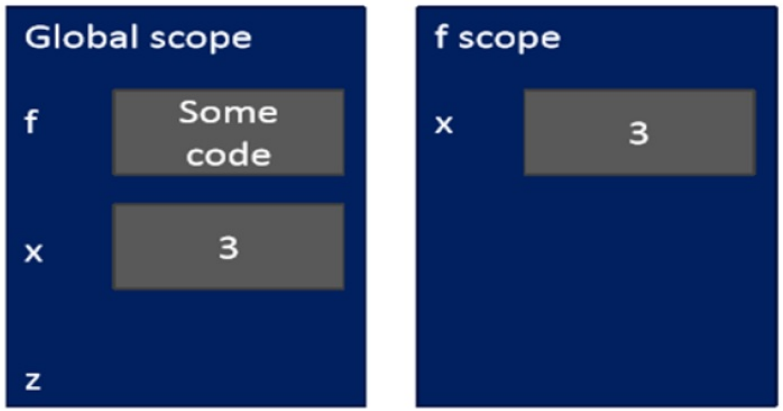

Figure 1: Değişken Kapsamı ve Fonksiyon

Değişken Kapsamı

```
def f( x ):
    x = x + 1
    print('in f(x): x =', x)
    return x

x = 3
z = f( x )
```

Figure 2: Değişken Kapsamı ve Fonksiyon

Değişken Kapsamı

```
def f( x ):
    x = x + 1
    print('in f(x): x =', x)
    return x

x = 3
z = f( x )
```

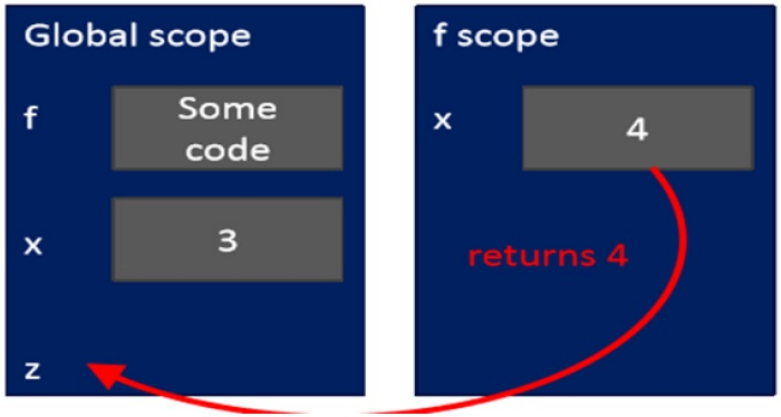

Figure 3: Değişken Kapsamı ve Fonksiyon

Değişken Kapsamı

```
def f( x ):
    x = x + 1
    print('in f(x): x =', x)
    return x

x = 3
z = f( x )
```

Figure 4: Değişken Kapsamı ve Fonksiyon

Uyarı return Olmayan İfade

Python, geri dönüş verilmezse None değerini döndürür None, NoneValue türündedir ve bir değerin yokluğunu temsil eder.

def is_even_one(val):

if val %2 ==0:

```
print('number is even')
```

else:

```
print('number is odd')
```

def is_even_two(val):

```
if val %2 ==0:

  return True
```

Uyarı return Olmayan İfade

```
r1 = is_even_one(4)
print('Value:',r1,'Type:',type(r1))
r2 = is_even_two(4)
print('Value:',r2,'Type:',type(r2))
r3 = is_even_two(5)
print('Value:',r3,'Type:',type(r3))
```

Uyarı return Olmayan İfade

Output:

number is even

Value: None Type: $<$ `class'NoneType'`

Value: True Type: $<$ `class'bool'`

Value: None Type: $<$ `class'NoneType'`

Özet- Bir İşlev Çağrıldığında

1. Gerçek parametreler değerlendirilir, biçimsel parametreler gerçek parametrelerin değerlerine bağlanır.
2. Yürütme noktası (kontrol), çağrı ifadesinden işlev içindeki ilk ifadeye geçer.
3. İşlevin (girintili) gövdesindeki kod, her ikisine kadar yürütülür
   - Bir return ifadesiyle karşılaşılır ve return ifadesinden sonraki değer döndürülür.
   - Fonksiyonun gövdesinde daha fazla ifade yoktur ve hiçbiri iade edilmez.
4. Yürütme noktası, çağrıyı takiben koda geri aktarılır.

Anahtar Kelime Argümanları

Şimdiye kadar, ilk biçimsel parametrenin birinci gerçek parametreye, ikincinin ikinciye vb. bağlı olduğu 'konumsal' argümanlar kullandık.

Ayrıca, biçimsel parametrelerin, biçimsel parametrenin adını kullanarak gerçek parametrelere bağlı olduğu anahtar kelime bağımsız değişkenlerini de kullanabiliriz.

Anahtar Kelime Argümanları

```
def printName(fistName,lastName,reverse):
if reverse:
print(lastName + ',' + firstName)
else:
  print(firstName, lastName)

printName('Joe','Smith',False)

printName('Joe','Smith',reverse=False)

printName('Joe',lastName='Smith',reverse =False)

printName(lastName='Smith', firstName= 'Joe',reverse =False
```

Anahtar Kelime Argümanları

Yukarıdaki ifadeler eşdeğerdir, hepsi printName öğesini aynı gerçek parametre değerleriyle çağırır. Anahtar sözcük argümanları herhangi bir sırada görünebilir, ancak anahtar sözcük argümanı bir anahtar sözcük argümanı, bu nedenle aşağıdakilere izin verilmez.

PrintName('Joe',lastName='Smith', False)

Varsayılan Değerler(Default Values)

Anahtar sözcük argümanları varsayılan parametre değerleriyle birlikte kullanılabilir.

def printName(firstName, lastName, reverse=False):

if reverse:

```
print(lastName + ',' + firstName)
```

else:

```
print(firstName, lastName)
```

Varsayılan değerler, programcıların bir işlevi belirtilen sayıdan daha az sayıda argümanlar.

Varsayılan Değerler(Default Values)

Aşağıdaki ifadeler:

printName('Joe', 'Smith')

printName('Joe', 'Smith', True)

printName('Joe', lastName = 'Smith', reverse = True)

Son ikisi anlamsal olarak eşdeğer olmak üzere çıktı verir:

Joe Smith Smith,Joe Smith,Joe

Argüman olarak Fonksiyonlar

Fonksiyon parametreleri (argümanlar), fonksiyonlar da dahil olmak üzere herhangi bir tipte olabilir. Bu, bir fonksiyonun başka bir fonksiyona parametre olarak aktarılabileceği anlamına gelir. Bir fonksiyon parametre olarak geçirildiğinde, fonksiyon çalıştırılmaz, fonksiyon referans işlev geçirilir.

Argüman olarak Fonksiyonlar

Örnek:

```
def funa(x):
#What is the difference between the following statements.
print(x)
print(x())
def funb():
return 50
funa(funb)
```

Output:

```
<function funb at 0x0000019DE1FC6730>
50
```

Argüman olarak Fonksiyonlar

Önceki kod örneğini aşağıdakiyle karşılaştırın, aradaki fark nedir?

```
def funa(x):
print(x)
def funb():
return 50
funa(funb())
```

Output

50

Argüman olarak Fonksiyonlar

```
def func_a():
print('inside func_a')
def func_b(y):
print('inside func_b')
return y
```

Argüman olarak Fonksiyonlar

```
def func_c(z):
print('inside_c')
return z ()
print(func_a())
print(5+ func_b(2))
print(func_c(func_a))
```

Argüman olarak Fonksiyonlar

Output:

inside func_a

None

inside func_b

7

inside func_c

inside func_a

None

Argüman olarak Fonksiyonlar

```
def func_a():
    print('inside func_a')
def func_b(y):
    print('inside func_b')
    return y
def func_c(z):
    print('inside func_c')
    return z()
print(func_a())
print(5 + func_b(2))
print(func_c(func_a))
```

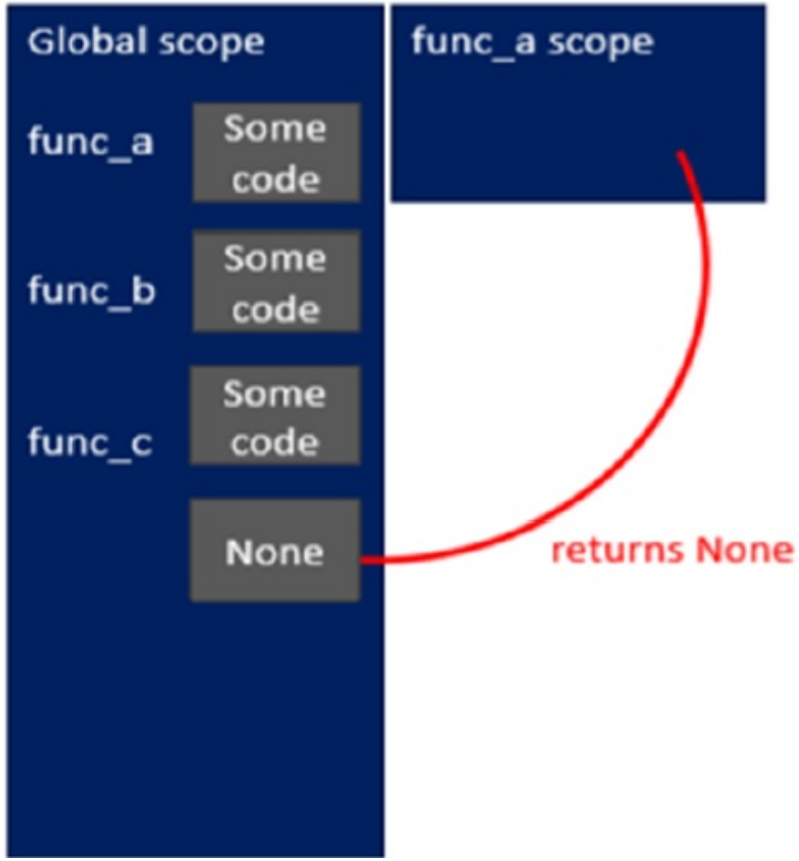

Figure 5: Fonksiyonlar

Argüman olarak Fonksiyonlar

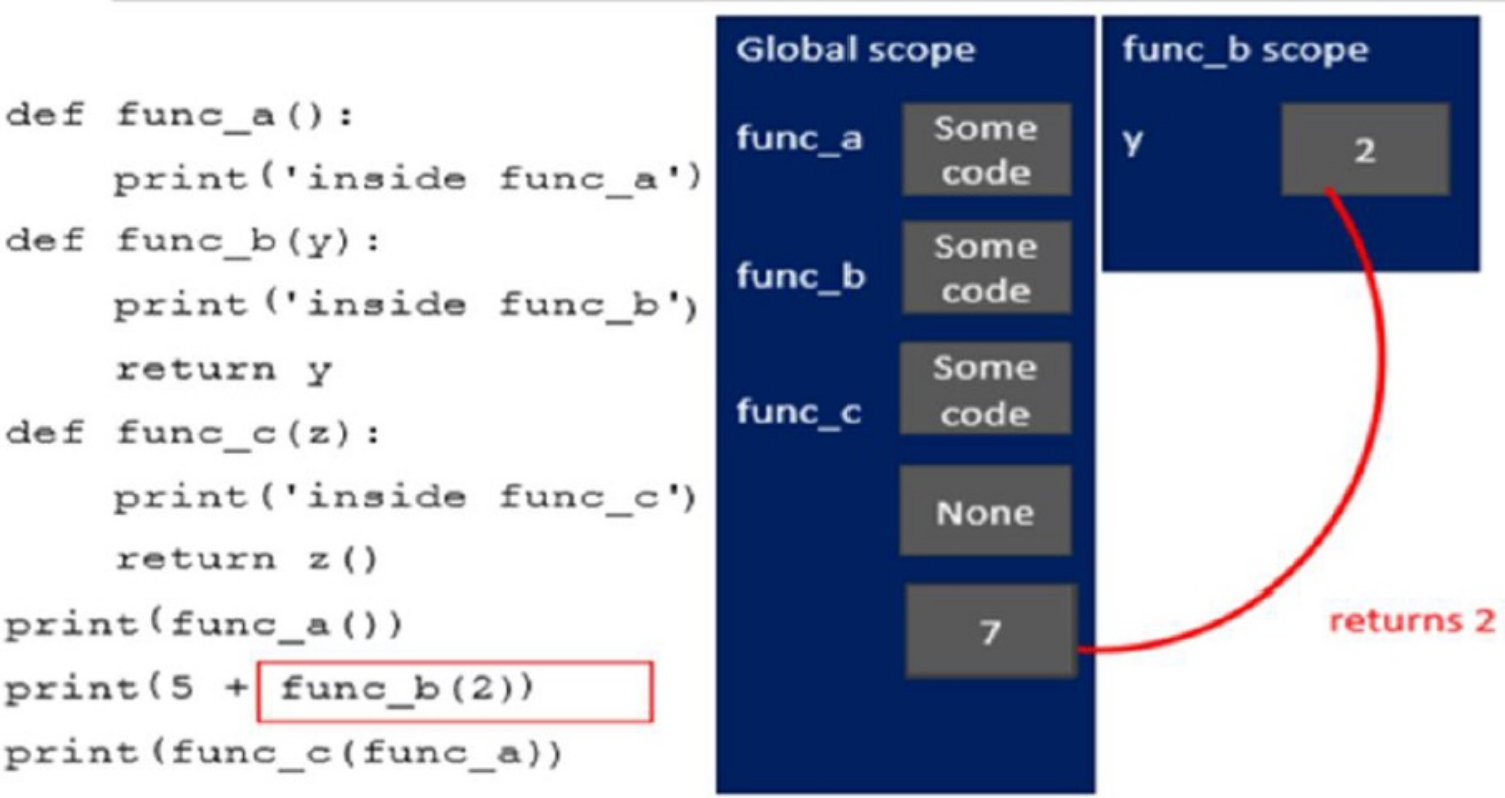

Figure 6: Fonksiyonlar

Argüman olarak Fonksiyonlar

```
def func_a():
    print('inside func_a')
def func_b(y):
    print('inside func_b')
    return y
def func_c(z):
    print('inside func_c')
    return z()
print(func_a())
print(5 + func_b(2))
print(func_c(func_a))
```

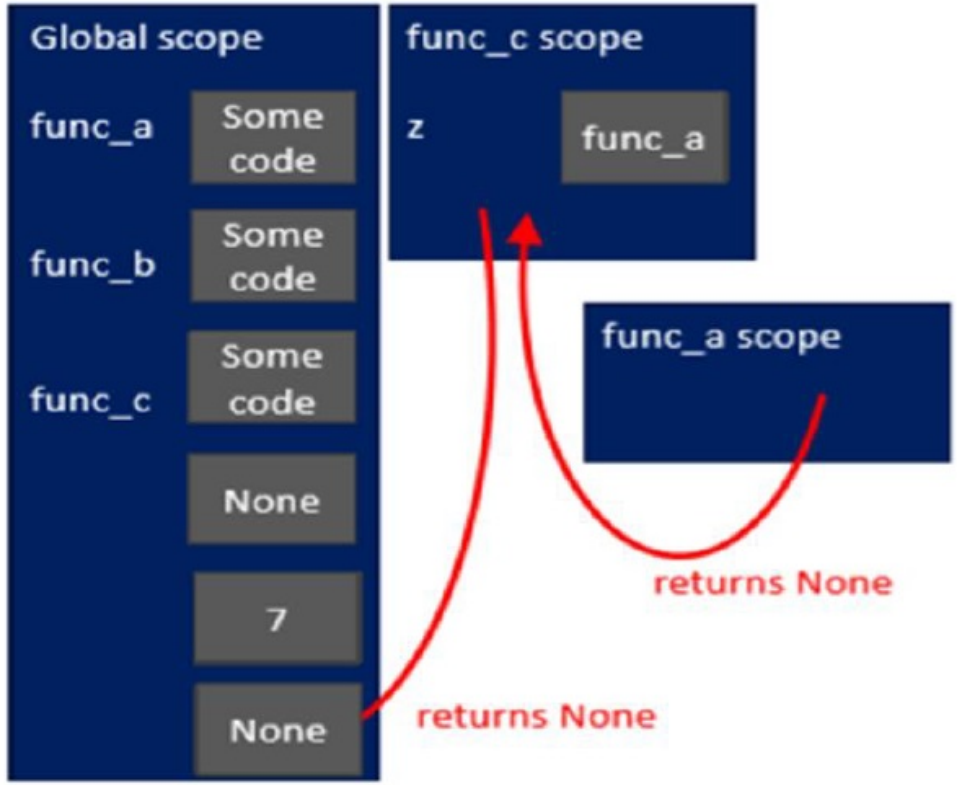

Figure 7: Fonksiyonlar

Kapsam Örneği

Bir fonksiyonun içinde, dışarıda tanımlanmış bir değişkene erişebilir.

Bir fonksiyonun içinde, dışarıda tanımlanmış bir değişkeni değiştiremez global değişkenler kullanabilirsiniz, ancak bu tavsiye edilmez.

Kapsam Örneği

```
def f(y):
x=1
x+=1
print(x)
x=5
f(x)
print(x)
```

Output: 2

5

Kapsam Örneği

```
def g(y):
print (x)
print(x+1)
x=5
g(x)
print(x)
```

Output:

```
5
6
5
```

Kapsam Örneği

```
def h(y):
x+=1
x=5
h(x)
print(x)
```

Kapsam Örneği

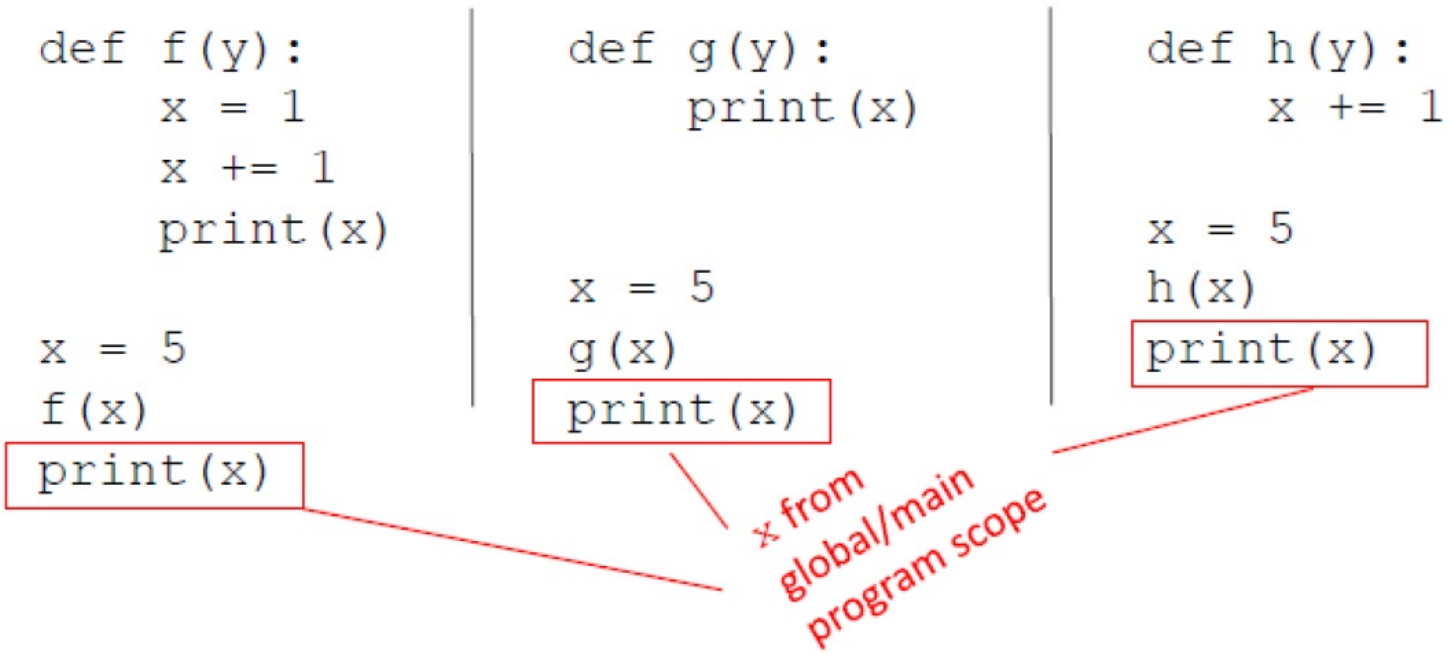

Figure 8: Kapsam

Kapsam Detayları

```
def g(x):
    def h():
        x = 'abc'
    x = x + 1
    print('g: x =', x)
    h()
    return x

x = 3
z = g(x)
```

Some code

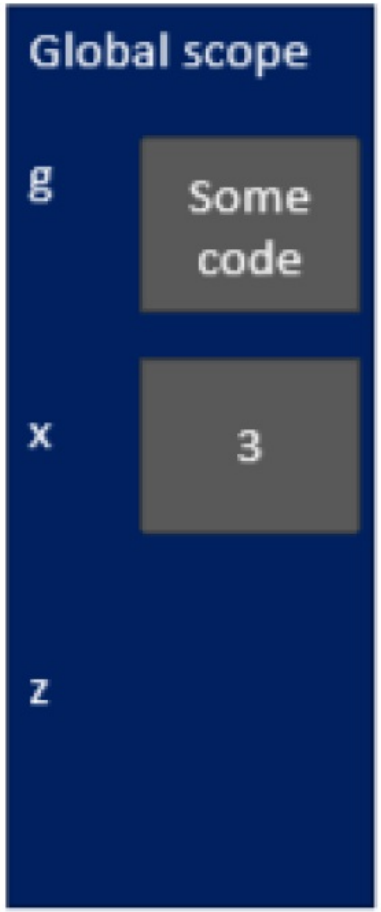

Figure 9: Kapsam Detayları

Kapsam Detayları

```
def g(x):
    def h():
        x = 'abc'
    x = x + 1
    print('g: x =', x)
    h()
    return x

x = 3
z = g(x)
```

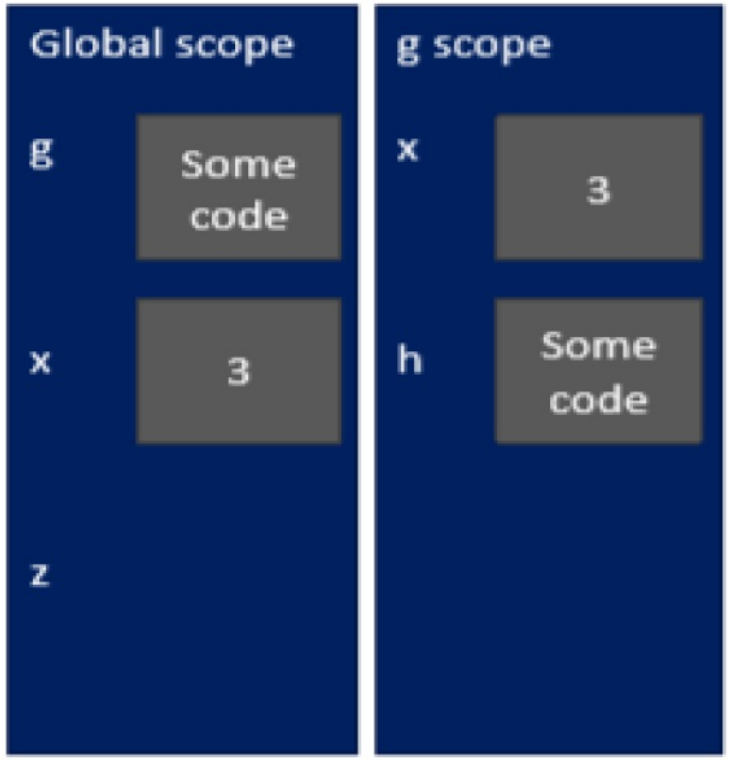

Figure 10: Kapsam Detayları

Kapsam Detayları

Figure 11: Kapsam Detayları

Kapsam Detayları

```
def g(x):
    def h():
        x = 'abc'
    x = x + 1
    print('g: x =', x)
    h()
    return x

x = 3
z = g(x)
```

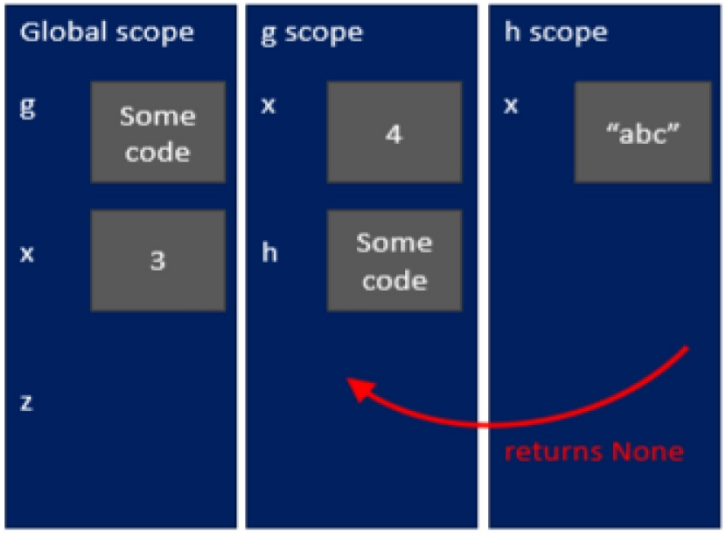

Figure 12: Kapsam Detayları

Kapsam Detayları

```
def g(x):
    def h():
        x = 'abc'
    x = x + 1
    print('g: x =', x)
    h()
    return x

x = 3
z = g(x)
```

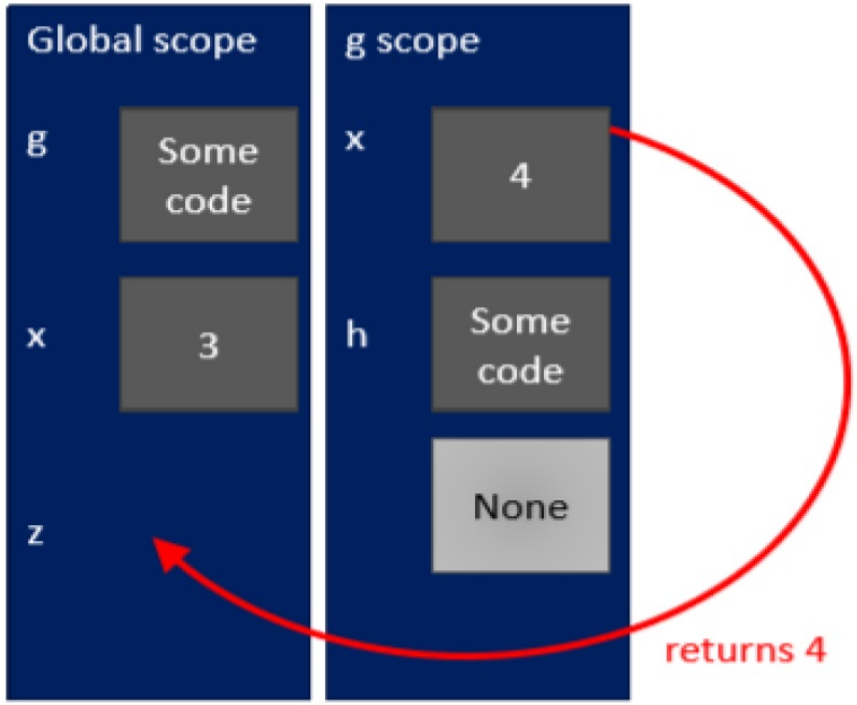

Figure 13: Kapsam Detayları

Kapsam Detayları

```
def g(x):
    def h():
        x = 'abc'
    x = x + 1
    print('g: x =', x)
    h()
    return x

x = 3
z = g(x)
```

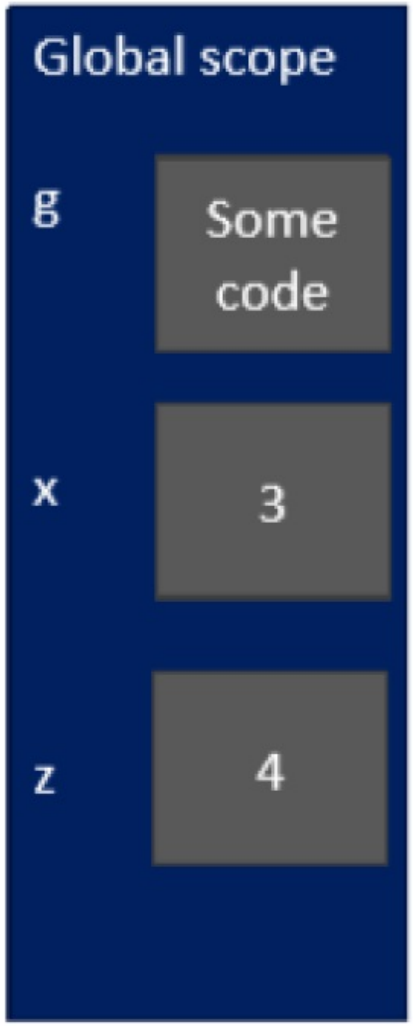

Figure 14: Kapsam Detayları

Aşağıdaki Programı İzleyin ve Çıktısı

```
def f(x):
def g():
x = 'abc'
print('x = ', x)
```

Aşağıdaki Programı İzleyin ve Çıktısı

```
def h():
z = x

print('z = ', z)

x = x + 1

print('x = ', x)

h()

g()

print('x = ', x)

return g
```

Aşağıdaki Programı İzleyin ve Çıktısı

```
x = 3
z = f(x)
print('x =', x)
print('z =', z)
z()
```

Aşağıdaki Programı İzleyin ve Çıktısı

Output:

x=4

z=4

x= abc

x=4

x=3

z= < functionf. < locals > .gat0x0000027C3EC98400 >

x=abc

Global Değişkenler

Python ayrıca global değişkenleri de destekler: fonksiyonların dışında tanımlanan değişkenler.

Bir global değişken tüm fonksiyonların içinden görülebilir.

Global bir değişkeni güncellemek isteyen herhangi bir fonksiyon global bildirim.

Fonksiyonların içinde değer atanan/güncellenen, global ile aynı isme sahip değişkenler değişkenleri, global bildirim olmadan global olarak kabul edilmeyecektir. Onlar olacak yerel değişkenler olarak kabul edilir.

Uyarı: Global değişkenler genellikle önerilmez. Bu bir çelişkidir. Değişkenlerin kullanılmadıkları fonksiyonlarda erişilebilir olmasını sağlamak için modülerlik. Küresel değişkenler beklenmedik sonuçlara neden olabilir.

Global Değişkenler Örneği

```
def f():
x = 1
print(x)
x = 5
f()
print(x)
```

Output:

```
1
5
```

Global Değişkenler Örneği

```
def g() :
global x
x = 1
print(x)
x = 5
g()
print(x)
```

Output:

```
1
1
```

Alıştırmalar

1. Pozitif bir tam sayının rakamlarının toplamını veren bir fonksiyon yazınız. Kullanıcı tarafından girilen pozitif sayıların rakamlarının toplamını bulmak için bu işlevi kullanın.
2. Bir sayıyı parametre olarak alıp o sayının tersini döndüren bir fonksiyon yazınız.
3. 2 tabanındaki bir ikili sayının belirli bir dizesini ondalık eşdeğerine dönüştüren ve ondalık değeri döndüren bir fonksiyon yazın.

- Öncelikle is_binary adında bir fonksiyon yazın; bu fonksiyon bir dizge parametresi alır ve string parametresi 0'lar ve 1'lerden oluşan bir ikili dizge ise True'yu (örneğin, '101'), aksi halde False'ı (örneğin, '123') döndürür.
- Daha sonra, bir dize parametresi alan ve parametre dizisinin ikili bir sayı olup olmadığını kontrol etmek için is_binary işlevinizi kullanan, Convert_to_decimal adlı bir işlev yazın.