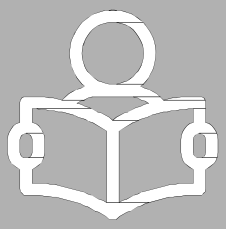
PYTHON

Fonksiyonlar, Global ve Lokal Değişkenler

Kapsam



- Fonksiyon Kullanımı
- Fonksiyon Oluşturma
- Fonksiyonlarda Parametre Türleri
- Değer Döndüren ve Döndürmeyen Fonksiyonlar
- Global ve Lokal Değişkenler

Fonksiyon Kullanımı

Bu bölüme kadar kullanılan fonksiyonlardan bazıları şunlardır:
print( ), input( ), int( ) ve len( )

Fonksiyonlar, matematikteki fonksiyonlarla aynı işlevi görürler, y=f(x).

Fonksiyonlar; kodları tekrar yazmayı ortadan kaldırmak, kodları daha iyi organize etmek sonrasında bu kodları rahat bir şekilde kullanmak için oluşturulur.

Python'da birçok fonksiyon çeşitli kütüphanelerde veya varsayılan olarak tanımlı gelir. Bu sayede en temel işlemler için bile satırlarca kodu tekrarlamaya gerek kalmaz.

Fonksiyon Kullanımı

Her fonksiyonun bir adı (print, input, len vb.) ve işlevini yerine getirmesi için kullanabileceği parametre adı verilen yapıları vardır.

Parametreler fonksiyonda tanımlı özellik ve girdileri belirtir.

Bir fonksiyonu kullanırken (çağırırken) bu parametrelere verilen değerlere “argüman” denir.

Fonksiyon Kullanımı

Aşağıda print( ) fonksiyonu ve parametrelerinin kullanımına örnek verilmiştir.
“end” ve “sep” birer parametredir.
Fonksiyonlarda bazı parametreler zorunludur ve bir argüman (değer) girmek gerekir.
Ancak yine bazı parametreler fonksiyon içinde ön tanımlı olarak bir değer/boş (None) veya seçmeli olarak belirlenmiştir.
Bu parametrelere değer vermek gerekmez.
Bunlar, isteğe bağlıdır ve kullanılmayacak özellikler için argüman girmeye gerek yoktur.


```
print ('Merhaba', 'Mars', sep=' ', end='\n') #parametreleri kullanarak
print ('Merhaba')
print ('Merhaba',` Mars') #parametreleri kullanmadan
print ('Merhaba')
```

Fonksiyon Oluşturma

Bir fonksiyon; "def fonksiyonAdi(parametre1, parametre2,..): " şeklinde tanımlanır.
Fonksiyon içindeki kodlar girinti şeklinde blok yapısında yazılır.
Fonksiyon yapısı dışında o fonksiyonu kullanmak için fonksiyonAdi(argüman1, argüman2) şeklinde parametre değerleri verilerek kullanılır.

Kod

```
def sayiCiftMi(sayi):
    if sayi%2==0:
        print('Sayı çifttir')
    else:
        print('Sayı tektir')
#fonksiyonun çağırılması
sayiCiftMi(10)
```

Fonksiyonun adı ve parametrelerine değerleri yazılarak fonksiyon kullanılabilir. Fonksiyon çağrıldığında verilen argümanlara göre işlem yapar. Örnekteki fonksiyon 10 sayısını kontrol ederek sonucu "Sayı çifttir" şeklinde ekrana yazdırır.

Örnek 1

Fonksiyon bir metni istenildiği kadar alt alta yazdırsın.
Bu fonksiyonda yazdırılacak metin ve yazdırılma sayısı olmak üzere iki adet parametre olacaktır.

Kod

```
def yazdir(metin,kacKere):
    for i in range (1, (kacKere+1)):
        print (metin, end='\n')
#Fonksiyon çağırma
yazdir('Merhaba', 5)
```

Örnek 2

Bir sayının asal bir sayı olup olmadığını bulan bir fonksiyon yazalım. Fonksiyon, sayı asal ise “True”;
değilse “False” değerini döndürecektir

Kod

```
def asalMi(sayi):
    sayac=2 # tüm sayılar 1'e bölüneceğinden 2 ile başlattık
    while sayac<=int(sayi/2):
       if sayi%sayac==0:
           return False
       sayac+=1
    return True
#Fonksiyonu çağırma
asalMi(113)
```

NOT

“for” döngüsünde sayının yarısına kadar bakmamızın nedeni bir sayının kendi değerinin yarısından önce böleni yok ise sonrasında da olmayacağı kuralıdır.

Örnek 3

Verilen sayının faktöriyelini bulan bir fonksiyon tanımlayalım.

Kod

```
def faktoriyelAl(sayi):
    sonuc=1
    if (sayi==0 or sayi==1):
        print('sonuç= ', 1)
    elif sayi>1:
        for i in range(1, sayi+1):
            sonuc*=i
        print ('sonuc=', sonuc)
    else:
        print ('0 veya daha büyük sayısal bir değer girmelisiniz')

faktoriyelAl(int(input('Faktöriyeli alınacak sayıyı giriniz: ')))
```

Fonksiyonlarda Parametre Türleri

"faktoriyelAl" olarak tanımlanan fonksiyon "sayi" parametresiyle çalışmaktadır. Argüman olarak bir sayı verildiğinde sayının faktöriyelini hesaplamaktadır. Fonksiyonlar yaptıkları işlemlere göre farklı parametre türlerini kullanır.

Kod

```
def agacCiz(agacinYuksekligi, karakter='*'):
    b=agacinYuksekligi
    for i in range(1,agacinYuksekligi+1):
        print(b*' ',(2*i-1)*karakter)
        b-=1
agacCiz(5)
agacCiz(5,"!")
```

Fonksiyonlarda Parametre Türleri

Kod

```
def agacCiz(agacinYuksekligi, karakter='*'):
    b=agacinYuksekligi
    for i in range(1,agacinYuksekligi+1):
        print(b*' ',(2*i-1)*karakter)
        b-=1
agacCiz(5)
agacCiz(5,"!")
```

Bu fonksiyonun "agacinyuksekligi" ve "karakter" adlarında iki adet parametresi bulunmaktadır. "agacinYuksekligi" ağacın satır sayısını belirten bir parametredir. Bu parametreye bir argüman (değer) verilmezse fonksiyon hata verir. Böyle parametrelere "zorunlu parametre" denir.
"karakter" parametresi ise parametre olarak tanımlanırken bir değerle birlikte tanımlanmıştır, bu değer parametrenin varsayılan değeridir. Varsayılan değer olarak "None" boş değer verilebilir. Zorunlu parametreler dışında fonksiyonlarda kullanılan diğer parametreler için varsayılan değerler tanımlanabilir. Eğer kullanıcı karakter olarak bir giriş yapmazsa yani boş bırakırsa fonksiyon ağacı varsayılan olarak "*" karakterini kullanarak oluşturur.

Değer Döndüren ve Döndürmeyen Fonksiyonlar

Python'da fonksiyonlardan bazıları sadece bir işlevi yerine getirir ve görevi orada biter.
Ama bazı fonksiyonlar bir değer döndürür.
"int( )" fonksiyonu verilen argümanı sayıya çevirerek döndürür.
Yukarıdaki örneklerde "faktoriyelAl" ve "agacCiz" fonksiyonları bir değer döndürmemektedir.
Değer döndüren fonksiyonlarda "return" ifadesi yer alır.
Bir fonksiyonun sadece ekrana yazı yazdırması değer döndürdüğü anlamına gelmez.
Eğer bir fonksiyonun çıktısı uygun bir değişkene atanabiliyorsa bu fonksiyon değer döndüren bir fonksiyondur.

Değer Döndüren ve Döndürmeyen Fonksiyonlar

Örnekte kullanılan faktoriyelAl() fonksiyonunu değer döndüren bir fonksiyon haline getirelim.

Kod

```
def faktoriyelAl(sayi):
    sonuc=1
    if (sayi==0 or sayi==1):
        sonuc=1
    elif sayi>1:
        for i in range(1, sayi+1, 1):
            sonuc*=i
    else:
        sonuc=-1 #hatalı bir işlem olduğunu anlamak için -1
değerini veriyoruz
    return sonuc

faktoriyelAl(4)
```

Örnek 4

Kendi tanımladığınız fonksiyonların içinde temel fonksiyonları ya da tanımladığınız başka fonksiyonları da kullanabilirsiniz.
Kullanıcıdan liste olarak aldığı sayıların ortalamasını bulan bir fonksiyon tanımlayabilir ve bunun içinde len( ) fonksiyonunu çağırabilirsiniz.

Kod

```
def ortalamaBul(sayilar):
    sayilarinToplami=0
    sayilarinOrtalamasi=0
    for i in range(len(sayilar)):
        sayilarinToplami+=sayilar[i]
    sayilarinOrtalamasi=sayilarinToplami/len(sayilar)
    return sayilarinOrtalamasi
```

NOT

Fonksiyonlar bir fonksiyon da dâhil olmak üzere değer olarak her türlü veri tipini döndürebilirler.

Global ve Lokal Değişkenler

Python'da blok yapısından ve girintilerden söz etmiştik. Python'da tanımlanan ve değer atanan değişkenler tanımlandıkları blok içinde geçerlidir. Bir değişken aynı düzeydeki veya daha içerideki girintilerde de değerini korur.

Kod

```
sayı1=0
for i in range (1,3):
    print ('i değişkenin değeri=', i)
print ('i değişkenin son değeri=', i)
```

Global ve Lokal Değişkenler

Kod

```
sayı1=0
for i in range (1,3):
    print ('i değişkenin değeri=', i)
print ('i değişkenin son değeri=', i)
```

"for" bloğu en dışta yer aldığı için i değeri onunla aynı hizada ve daha içerideki girintilerde tanınmakta ve değerini korumaktadır.
Ancak sadece iç girintideki bir blokta değer atanan değişken dış bloklarda tanımsızdır.
Bu tür değişkenlere yani kendi bloğu ve daha içerideki girintilerde tanınan değişkenlere yerel-local değişken denir.
En dışta tanımlanan (girintisiz) değişkenler global değişkenlerdir. Global değişkenler ilk değerlerini tüm alt bloklara taşırlar.
Alt bloklarda (iç girintide) aynı değişkene farklı bir değer atanabilir.

Örnek 5

Sayıların toplamını veren bir fonksiyon tanımlayalım.

Kod

```
topla=0 #global değişken
def toplamBul (sayiListesi):
    topla=0
    for i in range (len(sayiListesi)):
        topla+=sayiListesi[i]
    return topla
print(topla) #sıfır verir
toplamBul([1, 2, 3, 4, 5]) #15 verir
```

Fonksiyonda tanımlanmasına rağmen "topla" global değişkene atanan değer ekrana yazdırılabilir. Yani sıfır çıktısı alınır. Ama fonksiyon çağrılırsa işlem yapar.