Fonksiyonlar

Prof.Dr. Bahadır AKTUĞ
JFM212 Python ile Mühendislik Uygulamaları

*Kaynakça bölümünde verilen kaynaklardan derlenmiştir.

Fonksiyonlar

- Fonksiyonlar, sıkça tekrarlanan bir dizi işlemin yapılması için belirli sayıda parametre alan ve bunlarla işlem yaparak parametre değerlerini değiştiren veya bir değer döndüren kod parçacıklarıdır.
- Fonksiyonlar, farklı programlama dillerinde ve bazen işlevlerine göre (değer döndürüp döndürmemesi gibi) farklı isimler ve anahtar sözcükler kullanılmaktadır (subroutine, function, procedure vb.)
- Fonksiyon girdilerine parametre, bu parametrelerin değerlerine ise "argüman" denir.
- Fonksiyon parametrelerinin sayısı önceden bilinmiyorsa, değişken sayıda parametre alan bir fonksiyon da oluşturulabilir.

Fonksiyonlar

- Tüm programlama dillerinde olduğu gibi, Python'da da fonksiyonlar tanımlanabilir.
- Bazı programlama dillerinde fonksiyonların herhangi bir değer döndürmezken, diğer bazı programlama dillerinde ise değer döndüren/döndürmeyen fonksiyonlar farklı şekilde tanımlanmak zorundadır.
- Python'da fonksiyonlar değer döndürsün veya döndürmesin tek bir şekilde tanımlanırlar.
- Python'ın diğer birçok programlama dillerine göre bir avantajı da birden fazla değer döndürebilmesidir.
- Fonksiyon tanımlama genel olarak:

```
def fonksiyon(parametre1,parametre2,...):
```

şeklindedir.

Fonksiyonlar

- Bir fonksiyon hiçbir parametre de almayabilir. Ancak, öyle bile olsa fonksiyondan sonra parantezler bulunmak zorundadır.
- Örnek bir fonksiyon ve bu fonksiyonu çağıran komutlar aşağıdaki şekilde tanımlanabilir:

```
def ortalama(a,b,c):
    return (a+b+c)/3

x = 5
y = 6
z = 1
print('Üç sayının ortalaması: %4.2f' % (ortalama(x,y,z)))
Üç sayının ortalaması: 4.00
```

- Burada a,b ve c parametre, bu parametrelerin aldığı 5, 6, 1 değerler ise argümanlardır.

Seçimlik Parametreler

- Fonksiyon parametrelerinden bazıları seçimlik (optional) olabilir. Seçimlik parametre verilmediğinde varsayılan olarak bulunan argüman kullanılır.

```
# -*- coding=cp1254 -*-
def kayit(ad,milliyet="Türk"):
    print('%s-%s' % (ad,milliyet))

kayit('John Smith','İngiliz')
kayit('Ahmet DEMİR')
```

```
John Smith-İngiliz
Ahmet DEMİR-Türk
```

Anahtar Parametreler

- Seçimlik parametreler ile çalışırken, parametrelerin sırası önemlidir.
- Aşağıdaki örnekte hem "medenihal" hem de "milliyet" değişkenleri seçimliktir. Sonucun nasıl hatalı çıkabileceğine dikkat ediniz.

```
# -*- coding=cp1254 -*-
def kayit(ad,medenihal='Evli',milliyet="Türk"):
    print('%s-%s-%s' % (ad,medenihal,milliyet))
kayit('John Smith','İngiliz')
```

John Smith-İngiliz-Türk

Anahtar Parametreler

- Bu sorunu gidermek için seçimlik parametrelere çok benzer şekilde bir de anahtar parametreler vardır.
- Anahtar parametreler, fonksiyon tanımında bulunan parametre sırasından farklı olarak kullanılabilirler.

```
# -*- coding=cp1254 -*-
def kayit(ad,medenihal='Evli',milliyet="Türk"):
    print('%s-%s-%s' % (ad,medenihal,milliyet))

kayit('John Smith',milliyet='İngiliz')
```

```
John Smith-Evli-İngiliz
```

Çoklu Değer Döndürme

Python'da fonksiyonlar istenilen sayıda değer döndürebilirler:

```
def katlar(x):
  return 2*x,3*x,4*x,5*x,6*x

x2,x3,x4,x5,x6 = katlar(5)
print(x2,x3,x4,x5,x6)
```

```
10 15 20 25 30
```

Değişkenlerin Tanım Alanları (scope)

- Python'da değişkenlerin tanım alanları, tanımlandıkları alan ile sınırlıdır.
- Fonksiyon içinde tanımlanan değişkenlerin tanım alanları fonksiyon ile sınırlıdır.
- Aşağıdaki örnekte "metin" değişkeni içeriği ana programdan alınarak fonksiyon içinde yazdırılabilir.

```
def yaz():
  print(metin)

metin = 'JFM212'
yaz()
```

```
JFM212
```

Değişkenlerin Tanım Alanları (scope)

- Buna karşın, değişkene aşağıdaki şekilde fonksiyon içinde atama yapılıp atama öncesinde değişken kullanılırsa hata mesajı alınır.
- Hata mesajı, değişkenin değer atamasından önce kullanıldığını belirtmektedir.

```
def yaz():
  print(metin)
  metin = 'JFM211'

metin = 'JFM212'
yaz()
```

!Hata mesajı

Değişkenlerin Tanım Alanları (scope)

- Örneğin aşağıdaki örnekte, metin değişkenine fonksiyon içerisinde atama yapılıyor.
- Aşağıdaki program çalıştırıldığında aynı isimdeki değişkenlerin birbirlerine karışmadan kullanılabildiği görülmektedir.

```
def yaz():
  metin = 'JFM211'
  print(metin)

metin = 'JFM212'
yaz()
print(metin)
```

```
JFM211
JFM212
```

Değişkenlerin Tanım Alanları (scope)

- Ana programda bulunan ve fonksiyon içinde atama yapılan değişkeninin kullanılması için söz konusu değişken "global" olarak tanımlanır.

```
def yaz():
  global metin
  print(metin)
  metin = 'JFM211'
  print(metin)

metin = 'JFM212'
yaz()
print(metin)
```

Program çıktısı:

```
JFM212
JFM211
JFM211
```

Değişken sayıda fonksiyon parametresi

- Bazen, bir fonksiyona gönderilecek parametre sayısı her defasında farklı olabilir.
- Bu durumda, değişken adının önüne '*' koyarak değişkenin birden fazla parametre içerdiği (aslında bir demet (tuple)) belirtilebilir.

```
def ortalama(*degerler):
  return sum(degerler)/len(degerler)

print(ortalama(5,2,4,6))
print(ortalama(1,9,42,5,2,4,6))
```

Program çıktısı:

```
4.25
9.857142857142858
```

Parametre Değerlerinin Fonksiyon İçinde Değiştirilmesi

- Fonksiyona girdi olarak gönderilen değişkenlerin değerleri, "değiştirilebilir (mutable)" veya "değiştirilemez (immutable)" olmalarına göre fonksiyon içerisinde değiştirilebilir.
- Metin tipi değişken (string) değiştirilemez türde olduğundan fonksiyon içinde değiştirilemediğine dikkat ediniz.

```
def degistir(s):
  print(s)
  s = "JFM211"
  print(s)

s = "JFM212"
print(s)
degistir(s)
print(s)
```

Program çıktısı:

```
JFM212
JFM212
JFM211
JFM212
```

Parametre Değerlerinin Fonksiyon İçinde Değiştirilmesi

- Buna karşın, liste tipi değiştirilebilir (mutable) türde olduğundan fonksiyon içinde değiştirilebilir.

```
def degistir(s):
  print(s)
  s[0] = 14
  print(s)

s = [5,2,4,6]
print(s)
degistir(s)
print(s)
```

Program çıktısı:

```
[5, 2, 4, 6]
[5, 2, 4, 6]
[14, 2, 4, 6]
[14, 2, 4, 6]
```

Fonksiyonların Kendini Çağırması (Recursion/Özyineleme)

- Özellikle yinelenen fonksiyonlar için, fonksiyonların kendini çağırması gerekebilir.
- Bu işleme özyineleme (recursion) adı verilir.
- En basit örnekleri, faktöriyel ve fibonacci sayılarıdır.

```
def faktoriyel(sayi):
    if sayi == 1:
        return sayi
    else:
        return sayi * faktoriyel(sayi-1)

print(faktoriyel(5))
```

Program çıktısı:

```
120
```

Modüller ve Modüller İçinden Fonksiyonların Çağırılması

- Modüller, Python fonksiyon ve sınıflarının bulunduğu dosyalardır.
- Özellikle büyük programlarda, fonksiyon ve sınıfları organize etmek için birbirleriyle ilişkili fonksiyon ve sınıflar farklı isimlerle modüllerde birleştirilebilir.
- Başka bir modüldeki sınıf veya fonksiyonu kullanabilmek için:
  - Python'ın arama dizininde ilgili modül bulunmalıdır.
  - «import» fonksiyonu ile ilgili modül programa dahil edilmelidir.
- «import fonksiyonu kullanılırken, bir modüldeki tüm fonksiyonlar birden alınabildiği gibi, sadece gerekli olanlar da isimleriyle alınabilir.
  - import dosya → dosya isimli modül
  - from dosya import fonksiyon → dosya isimli modüldeki «fonksiyon» isimli fonksiyon
  - from dosya import * → dosya isimli modüldeki tüm fonkisyon ve sınıflar
- Kullanacağımız modülün Python arama dizininde olup olmadığını aşağıdaki şekilde kontrol edebiliriz:
  - >>> import sys
  - >>> print(sys.path)

Modüller ve Modüller İçinden Fonksiyonların Çağırılması

matematik.py

```
def faktoriyel(sayi):
    if sayi == 1:
        return sayi
    else:
        return sayi *
faktoriyel(sayi-1)
```

ana program

```
from matematik import faktoriyel
print(faktoriyel(5))
```

Kaynakça

1 Wentworth, P., Elkner, J., Downey, A.B., Meyers, C. (2014). How to Think Like a Computer Scientist: Learning with Python (3nd edition).
2 Pilgrim, M. (2014). Dive into Python 3 by. Free online version: DiveIntoPython3.org ISBN: 978-1430224150.
3 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
4 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
5 Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.
6 Millman, K.J., Aivazis, M. (2011). Python for Scientists and Engineers, Computing in Science & Engineering, 13, 9-12.
7 John D. Hunter (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95.
8 Travis E. Oliphant (2007). Python for Scientific Computing, Computing in Science & Engineering, 9, 10-20.
9 Goodrich, M.T., Tamassia, R., Goldwasser, M.H. (2013). Data Structures and Algorithms in Python, Wiley.
10 http://www.diveintopython.net/
11 https://docs.python.org/3/tutorial/
12 http://www.python-course.eu
13 https://developers.google.com/edu/python/
14 http://learnpythonthehardway.org/book/