Open Source Coding: Python

Bölüm 3: Deyimler (II), Fonksiyonlar

Python'da Deyimler

Deyim	Rol	Örnek
Atama	Referans oluşturmak	`a, b = 'bir', 'iki'`
Çağrılar	Fonksiyon çalıştırmak	`log.write("something")`
if / elif / else	Mantıksal seçim	`if "python" in text:` `  print("yes!")`
for / else	Tekrarlama	`for x in mylist:` `  print(x)`
while / else	Genel döngüler	`while X > Y:` `  X = X + 1`
pass	Boş yertutucu	`while True:` `  pass`
break	Döngüden çıkmak	`while True:` `  if test(): break`
continue	Döngüde atlamak	`while True:` `  if skip(): continue`
def	Fonksiyon tanımlamak	`def f(a, b, c=1, d*):` `  print(a + b + c + d[0])`
return	Fonksiyon sonucu	`def f(a, b, c=1, d*):` `  return(a + b + c + d[0])`
global	Namespace	`global x`
nonlocal	Namespace (3.x)	`nonlocal x`

Python’da Deyimler

Deyim	Rol	Örnek
import	Modül erişimi	`import sys`
from	Nitelik erişimi	`from sys import stdin`

class	Sınıf yaratmak	`class Subclass(Superclass):`
		`  staticData = []`
		`  def method(self): pass`

try/except/finally	Hata yakalama	`try:`
		`  action`
		`except: print(‘oops’)`

raise	Hata fırlatma	`raise EndSearch(‘location’)`
assert	Test etmek	`assert X > Y,`
with/as	Context yönetimi	`with open(‘data’) as myfile:` `  process(myfile)`
del	Referans silme	`del data[k]` `del obj.attr`

Tekrarlanabilir (Iterable) Nesneler

- Tekrarlama: Bir dizinin nesnelerini sırayla ziyaret etmek
- Tekrarlanabilir olmak: Iterable protokolünü desteklemek
- Bazı tekrarlanabilir nesneler:
  - **List**: [0, 1, “str”, 4]; **Tuple**: (True, False, 5)
  - **Dictionary**: {‘key’:’value’, ‘a’:2}
  - **Set**: {‘x’, ‘y’, ‘z’} …

for döngüsü

- Alışılagelmiş for döngülerinden farklı, daha **genel**
- Diğer dillerdeki **for each** gibi
- Iterable nesnelerin elemanları arasında ilerler

```
$ python3
>>> for i in [1,2,3,4]:
...   print(i)
...
1
2
3
4
```

List'te Tekrarlama

```
$ python3
>>> numbers = []
>>> while True:
...     a = int( input('> ') )
...     if a == -1:
...         break
...     numbers.append(a)
...   - - - - - - - - - -
>>> sum = 0
>>> for i in numbers:
...     sum += numbers[i]
...
>>> avg = sum / len(numbers)
```

Dictionary'de Tekrarlama (i)

```
$ python3
>>> track = dict(name='Maxwell\'s Silver
Hammer', artist='The Beatles',
album='Abbey Road', genre='rock')

>>> for key in track:
...     print(key + ': ' + track[key])
...
album: Abbey Road
name: Maxwell's Silver Hammer
genre: rock
artist: The Beatles
```

Dictionary'de Tekrarlama (ii)

```
>>> for key in track.keys():
...     print(key + ': ' + track[key])
...
album: Abbey Road
name: Maxwell's Silver Hammer
genre: rock
artist: The Beatles

>>> for value in track.values():
...     print(value)
...
Abbey Road
Maxwell's Silver Hammer
rock
The Beatles
```

Daha tanıdık bir for döngüsü

```
$ python3
>>> L = [1, 2, 6, 24, 120]
>>> for i in range(0,5):
...     print(L[i])
...
1
2
6
24
120
```

Java:

```
int[] L = {1, 2, 6, 24, 120};
for(int i=0; i<5; i++) {
  System.out.println(L[i]);
}
```

Range (i)

```
$ python3
>>> S = 'abcdefghijk'
>>> for i in range(len(S)):
...   print(i, S[i])
...
0 a
1 b
2 c
3 d
4 e
5 f
6 g
7 h
8 i
9 j
10 k
```

Range (ii)

```
$ python3
>>> S = 'abcdefghijk'
>>> for i in range(0, len(S), 2):
...     print(i, S[i])
...
0 a
2 c
4 e
6 g
8 i
10 k
>>> list(range(0, len(S), 2))
[0, 2, 4, 6, 8, 10]
```

Range (iii)

```
$ python3
>>> L = [ x**2 for x in range(0,10) ]
>>> L
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> L = [x**2 for x in range(0,10)
... if x%2 == 0]
>>> L
[0, 4, 16, 36, 64]
```

Zip (i)

```
$ python3
>>> L1 = [1, 2, 3, 4]
>>> L2 = [5, 6, 7, 8]

>>> list(zip(L1, L2))
[(1,5), (2,6), (3,7), (4,8)]

>>> for (x, y) in zip(L1, L2):
...     print(x, y, '_', x+y)
...
1 5 _ 6
2 6 _ 8
3 7 _ 10
4 8 _ 12
```

Zip (ii)

```
$ python3
>>> T1, T2, T3 = (1,2,3),(4,5,6),(7,8,9)
>>> list(zip(T1, T2, T3))
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

```
>>> S1 = ‘abc’
>>> S2 = ‘xyz123’

>>> list(zip(S1, S2))
[(‘a’,’x’), (‘b’,’y’), (‘c’,’z’)]

>>>> dict(zip(S1, S2))
{'c': 'z', 'b': 'y', 'a': 'x'}
```

Enumerate

```
$ python3
>>> S = ‘python’
>>> for (offset, item) in enumerate(S):
...     print(item, ‘at ‘, offset)
...
p at 0
y at 1
t at 2
h at 3
o at 4
n at 5
```

Max ve Min

```
$ python3
>>> L = [0, 4, 1, 9, 11]
>>> max(L)
11
>>> min(L)
0

>>> S = ‘abczyx’
>>> max(S)
‘z’
>>> min(S)
‘a’
```

Fonksiyonlar

- **Yeniden kod kullanımı**
  - Birden fazla defa çalıştırılabilen,
  - Bir değer döndürebilen
- deyim grupları
- Python'daki en basit program yapısı

Fonksiyon Tanımlama

- **def**: Yeni bir nesne yaratıp, onu bir isime atamak

- “Python’da her şey nesnedir”

```
$ python3
>>> def printer(message):
...     print( ‘mesaj: ‘ + str(message) )
...
>>> printer(‘merhaba’)
mesaj: merhaba

>>> def topla(a, b):
...     return a + b
...
>>> topla(2, 3)
5
```

Değişkenlerin Kapsamı (i)

- Bir fonksiyonun içinde tanımlanan değişkenler, sadece o fonksiyonun içinde tanımlıdır.

- **local**

```
>>> def fn():
...     a = 5
...
>>> a
NameError: name 'a' is not defined
```

Değişkenlerin Kapsamı (ii)

- Varsayılan olarak, bir fonksiyonun içinde tanımlanan değişkenler, dışarıda tanımlanan değişkenlerle çakışmazlar.

```
>>> a = 10       # Global kapsamda a
>>> def fn():
...     a = 5  # Local kapsamda a
...
>>> a
10
```

Değişkenlerin Kapsamı (iii)

- Tüm fonksiyonların dışında tanımlanan değişkenler, o dosyada her yerden erişilebilir.
- Ancak fonksiyon içinde değerleri değiştirildiğinde, bu fonksiyon dışına yansımaz.
- **global**

```
count = 0
def fn1():
   print(count + 2)
def fn2():
   print (count + 3)
fn1()
fn2()
```

```
2
3
```

İç İçe Fonksiyonlar

- İzolasyon

```
def avg(L):
    def sum(L):
        s = 0
        for  item in L:
            s += item
        return s
    return '%.3f' % (sum(L) / len(L))

L = [5, 0, 47, 9, 2, 10]
print( avg(L) )
```

```
12.167
```

LEGB Kuralı

- İsim refersansları:
  - Local
  - Enclosing (kapsayan)
  - Global ve nonlocal tanımlanmış değişkenler
  - Built-in

sırasına göre aranır ve kullanılır.

global Deyimi

- Fonksiyon dışı kapsamdaki değişkene ulaşmayı sağlar.

```
X = 88

def fn():
      X = 99
      print('X in fn: '
      + str(X))

fn()

print('X in global
scope: ' + str(X))
```

```
X in fn: 99
X in global scope: 88
```

```
X = 88

def fn():
      global X
      X = 99
      print('X in fn: '
      + str(X))

fn()
print('X in global
scope: ' + str(X))
```

```
X in fn: 99
X in global scope: 99
```

Daha İyi Dizayn - Yan Etkilerden Kurtulmak

- **global** deyimini mümkün olduğunca az kullanmak

```
X = 99

def func1():
    global X
    X = 88
def func2():
    global X
    X = 77
```

İç İçe Fonksiyonlarda Kapsam

```
X = 99
def f1():

    X = 88

    def f2():
        print(X)

    f2()
f1()
```

```
88
```

nonlocal Deyimi (Python 3.x)

- İç fonksiyondan, dış fonksiyon kapsamındaki değişkene ulaşmayı sağlar.

```
X = 10
def outer():
    X = 15
    def inner():

        X = 20
        print('inner X:' +
        str(X))
    inner()
    print('outer X:' +
    str(X))
outer()
```

```
inner X: 20
outer X: 15
```

```
X = 10
def outer():
    X = 15
    def inner():
        nonlocal X
        X = 20
        print('inner X:' +
        str(X))
    inner()
    print('outer X:'+
    str(X))
outer()
```

```
inner X: 20
outer X: 20
```

Argümanlar ve Referanslar

- Mutable / immutable farkı:

```
def changer(a, b):
   a = 2
   b[0] = 'bir sey'
```

```
X = 1
L = [1, 2]
changer(X, L)
print(X, L)
```

```
1 ['bir sey', 2]
```

Tuple İle Birden Fazla Değer Döndürmek

```
def multiple(x, y):
   x = 2
   y = [3, 4]
   return x, y

X = 1
L = [1, 2]
X, L = multiple(X, L)

print((X, L))
```

```
(2, [3, 4])
```

Değişken Sayıda Argüman

```
def sth(a, *b):
  print(a)
  print(type(b))
  print(b + (a,))

sth(2, [3,4], 5, 6, 7, 8, 9, 10)
```

```
2
<class 'tuple'>
([3, 4], 5, 6, 7, 8, 9, 10, 2)
```

Argümanları Dağıtmak

```
def strConcat(a, b, c, d):
  return a + b + c + d

args = ('x', 'y', 'z', 't')

concatenated = strConcat(*args)

print(concatenated)
```

```
xyzt
```

Anahtar Kelimeli Argümanlar

```
def fn(first, second, third):
  print(first, second, third)

fn(third=5, first=42, second=10)
```

```
42, 10, 5
```

Anahtar Kelimeli, Değişken Sayılı Argümanlar

```
def printAgain(a, **args):
  print(type(args))
  print(args)

printAgain(0, day=4, month=3, year=2014)
```

```
<class 'dict'>
{'day': 4, 'year': 2014, 'month': 3}
```

Recursion (Özyineleme)

```
def factorial(n):
   if n <= 1:
      return 1
   return n * factorial(n-1)

print(factorial(6))
```

```
720
```

```
6 * 5 * 4 * 3 * 2 * 1
```

Dosya İşlemleri: Dosyayı Açmak

- `afile = open(filename, mode)`
- mode:
  - **r**: read
  - **w**: write
  - **a**: append (sonuna ekle)

Dosya İşlemleri: Dosyadan Okumak (i)

- En basit okuma:

```
contents = open('file.txt','r').read()
```

- Satır satır okuma:

```
myfile = open('file.txt','r')
row = myfile.readline()
print(row)
```

```
this is the first line of file.txt
```

Dosya İşlemleri: Dosyadan Okumak (ii)

- File iterator ile döngü:

```
for line in open('file.txt'):
        print(line, end='')
```

```
line1
line2
…
```

Dosya İşlemleri: Dosyaya Yazmak

```
myFile = open('file.txt','w')
myFile.write('dosyaya ')
myFile.write('yazmak bu kadar kolay')
myFile.close()
```

Alıştırma 1: Döngülerle çalışmak

- Kullanıcıdan, sırayla birden fazla kişi için bilgiler girmesini isteyelim, yaşı en büyük olan kişinin ya da kişilerin adını bulalım.

- kişi bilgileri: isim, soyisim, yaş

- Bir kişiyi bir **Dictionary**'de, tüm kişileri de bir **List**'te tutalım.

- **While** döngüsü kullanarak kullanıcının istediği kadar kişi kaydedelim, **for** döngüsü ve iterable kullanarak en yaşlı kişiyi/ kişileri bulalım.

```
Bir kişi ismi girin (-1=çık): Oğuz
Bu kişi için bir soyisim girin: Bilgener
Bu kişi için bir yaş girin: 18
- - - - - - -
En yaşlı kişi: Abdullah Atalar
```

Alıştırma 2: Fonksiyonlarla çalışmak

- Bir String'deki karakterlerin alfebatik değerlerinin toplamını veren bir fonksiyon yazalım ve bu fonksiyonu deneyelim.

- alfebatik değer: a = 1, b = 2, z = 26

- İpucu: **ord('a') = 97, ord('z') = 122, ord('A') = 65**

Alıştırma 3: Dosyaya yazmak

- Alıştırma 1'de elde ettiğimiz kişileri dosyaya yazalım

Teşekkürler!

Bir sonraki oturumda görüşmek üzere.