Mantıksal Kontrol ve Döngü Komutları

Prof.Dr. Bahadır AKTUĞ
JFM212 Python ile Mühendislik Uygulamaları

*Kaynakça bölümünde verilen kaynaklardan derlenmiştir.

Mantıksal Karşılaştırmalar

- Python, koşullu karşılaştırma için hemen hemen bütün programlama dillerinde olduğu gibi, "if/elif/else" bloklarını sunar.
- "If" komutundan sonra "True" veya "False" olarak yorumlanabilecek "boolean" bir ifade yer almalıdır.
- Boolean ifadeler:
- Aşağıdaki ifadeler "False" olarak alınır:
  - Sıfır değerli tüm sayılar
  - False değeri,
  - Boş metin değişkenleri,
  - Boş listeler/demetler/sözlükler
  - None değeri
- Bunların dışındaki her türlü değer/değişken "True" olarak kabul edilir.

if / elif / else

"If" komutunun genel hali aşağıdaki gibidir. "elif" komutu istenildiği kadar kullanılabilir.

```
if kosul-1:
    Komut Bloğu
elif kosul-2:
    Komut Bloğu
elif kosul-3:
    Komut Bloğu
else:
    Komut Bloğu
```

Ternary if

Ternary "if" üç operant alan bir komuttur. Komutun genel hali aşağıdaki gibidir.

```
max = a if a > b else b
```

Bu komutu normal "if" ile aşağıdaki şekilde de yazabiliriz.

```
if a > b:
    max=a
Else:
   max=b
```

Döngüler

- Python'da döngüler doğrudan diziler üzerinde (listeler, demetler, sözlükler, metin tipi değişkenler) veya while/for yapıları ile yapılır.
- Belirli bir sayıda döngü oluşturulacağı zaman "for" komutu veya diziler tercih edilir.
- Belirli bir koşula bağlı olarak döngü yapılacaksa, "while" komutu tercih edilir.
- Her iki komut da içiçe (nested) yapı şeklinde kullanılabilir.
- Her iki döngü için de ayrıca kontrol komutları vardır:
  - continue
  - pass
  - break
- Üzerinde döngü oluşturulan bir dizinin elemanları da bir dizi şeklindeyse, birden fazla döngü değişkeni kullanılabilir.

Döngüler

Doğrudan diziler ile:

Listeler, demetler, kümeler, sözlükler ve metin tipi değişkenler ile döngü oluşturulabilir.

```
>>> renkler = set(["Sarı", "Mavi", "Yeşil"])
>>> for renk in renkler:
...         print(renk)
...
Sarı
Mavi
Yeşil
```

Döngüler

Doğrudan diziler ile:

Listeler, demetler, kümeler, sözlükler ve metin tipi değişkenler ile döngü oluşturulabilir.

```
>>> ders = "JFM212"
>>> for harf in ders:
...     print(harf)
...
J
F
M
2
1
2
```

Döngüler için yararlı fonksiyonlar

range()

- Range komutu, belirlenen aralık dahilinde tamsayılardan oluşan bir "range" nesnesi üretir.
- Bu nesne, istenilen dizi türüne (liste, demet vb.) dönüştürülebilir.
- Komutun genel hali:

range([başlangıç], bitiş[, artım])

- "range" komutunun parametreleri tamsayı olmalıdır.
- Python versiyon 3'den itibaren "range" komutu bir "iterator" nesnesi döndürmektedir.

Döngüler için yararlı fonksiyonlar

range() örnekleri:

```
>>> for i in range(5):
...     print(i)
...
0 1 2 3 4

>>> list(range(0,10,3))
[0, 3, 6, 9]

>>> list(range(5,10))
[5, 6, 7, 8, 9]
```

Döngüler için yararlı fonksiyonlar

enumerate()

- "enumerate" komutu, girdi olarak verilen dizinin elemanları için sıralı tamsayı endeks değerleri üretir.
- "enumerate" komutu aralık dahilinde tamsayılardan oluşan bir "enumerate" nesnesi üretir.
- Bu nesne, istenilen dizi türüne (liste, demet vb.) dönüştürülebilir. Komutun genel hali:

- enumerate(dizi [, başlangıç endeksi=0])

- "enumerate" komutunun parametreleri tamsayı olmalıdır.

Döngüler için yararlı fonksiyonlar

enumerate() örnekleri:

```
>>> list(enumerate(choices))
[(0, 'döner'), (1, 'adana'), (2, 'iskender'), (3, 'mantı')]

>>> for index, item in enumerate(choices, start = 1):
...     print(index,item)
...
1 döner
2 adana
3 iskender
4 mantı
```

Döngüler için yararlı fonksiyonlar

zip()

- "zip" fonksiyonu birden fazla diziyi alarak elemanlarını sıralı olarak eşleştirir.
- "zip" komutu girdi dizilerin boyutları dahilinde bir "zip" nesnesi üretir.
- Bu nesne, istenilen dizi türüne (liste, demet vb.) dönüştürülebilir. Komutun genel hali:

- zip(a, b [, c, d, …])

Döngüler için yararlı fonksiyonlar

zip() örnekleri

```
>>> zip(range(5), range(1,20,2))
[(0, 1), (1, 3), (2, 5), (3, 7), (4, 9)]

>>> colors = ['red', 'green', 'blue']
>>> vals = [55, 89, 144, 233]
>>> for col, val in zip(colors, vals):
...         print(col, val)
('red', 55)
('green', 89)
('blue', 144)
```

Döngüler

for:

Belirli bir sayıda döngü oluşturulmak istendiğinde, aşağıdaki şekilde kullanılır.

```
>>> for i in range(5):
...     print(i)
...
0
1
2
3
4
```

Döngüler

while:

- Döngünün sonlandırılması belirli bir koşula bağlı olduğunda, "while" yapısı tercih edilir.
- "while" komutundan sonra "True" veya "False" olabilen "boolean" bir ifade bulunmalıdır.

```
>>> n = 3
>>> i = 0
>>> while i < n:
...   print(i)
...   i += 1
...
0
1
2
```

Döngü içi Komutlar:

break

- Döngünün (hem while hem de for) herhangi bir nedenle sonlandırılması için kullanılır.
- İçiçe (nested) döngüler olması durumunda en içteki döngü sonlandırılır.

```
>>> for harf in "JFM212":
...     if harf == 'M':
...         break
...     print(harf)
...
J
F
```

Döngü içi Komutlar:

continue

- Döngünün (hem while hem de for) herhangi bir nedenle döngünün başına gidilmesi için kullanılır.
- continue komutu kullanıldığında, döngü bir sonraki elemandan devam eder.

```
>>> for harf in "JFM212":
...     if harf == 'F':
...         continue
...     print(harf)
...
J
M
2
1
2
```

Döngü içi Komutlar:

pass

- Döngü içinde bir blokta komut kullanılması gerekiyor ama herhangi bir işlem yapılması istenmiyorsa (hem while hem de for) "pass" komutu kullanılır.

```
>>> for harf in "JFM212":
...   if harf == 'M':
...       pass
...   print(harf)
...
J
F
M
2
1
2
```

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/