Open Source Coding: Python

Bölüm 2: Veri Türleri ve Deyimler (I)

Python'da Program Yapısı

- Programlar **modül**lerden oluşur.
- Modüller **deyim**leri (statement) içerir.
- Deyimler **ifade**leri (expression) içerir.
- İfadeler **nesne**leri oluşturur ve işler.

Python’da Veri Türleri (Nesneler)

Numbers	`42, 5.712, 0b111, Decimal(), Fraction()`
String	`‘linux’, “python”, b’a\x01c’, u’sp\xc4m’`
List	`[1, [2, ‘lol’], 3.4], list(range(10))`
Dictionary	`{‘food’: ‘hamburger’, ‘type’:’whopper’}, dict(hours=10)`
Tuple	`(1, 4, ‘hello’, 2.7), tuple(‘asd’),`
File	`open(‘file.txt’), open(r’C:\Users\oguz \file.txt’, ’wb’)`
Set	`set(‘abc’), {‘a’, ‘b’, ‘c’}`
Diğer	`Boolean, None, type`

Numerik Veri Türleri

1. Tam sayılar (integers)
2. Ondalıklı sayılar (floating-point)
3. Karmaşık sayılar
4. Decimal (fixed precision floating point)
5. Fraction (kesir)
6. Set
7. Boolean

Integer

- int, long yerine **tek tip**
- Onluk sayılar: 123
- Onaltılık sayılar: 0xabc01
- Sekizlik sayılar: 0o1234
- İkilik sayılar: 0B0 0b1

Floating Point

- 572.2123
- CPython'da ondalıklı sayılar C'deki double veri türüyle ifade edilir.

Karmaşık Sayılar

- gerçek + sanal**j**
- 3 + 4j
- complex(3, 4)

```
$ python3
>>> 1j * 1J
(-1+0j)
>>> 2 + 1j * 3
(2+3j)
>>> (2 + 1j) * 3
(6+3j)
```

Decimal

```
$ python3
>>> 0.1 + 0.1 + 0.1 - 0.3
5.551115123125783e-17

>>> from decimal import Decimal

>>> Decimal('0.1') + Decimal('0.1') +
Decimal('0.1') - Decimal('0.3')
Decimal('0.0')

>>> Decimal(1) / Decimal(7)
Decimal('0.14285714285714285714285714285')
```

Fraction

```
$ python3
>>> from fractions import Fraction
```

Fraction (**pay**, **payda**)

```
>>> x = Fraction(1,3)

>>> Fraction(2,6) # Sadeleştirme
Fraction(1,3)
>>> print(x)
1/3
>>> Fraction('1.25')
Fraction(5, 4)
```

Matematiksel İşlemler

- Temel işlemler:
  - +, -, *, /, %
- Python'a özgü işlemler:
  - **, //
- Yardımcı metodlar:
  - pow, abs, round, int, bin…
- math modülü

Kıyaslamalar

- ==, !=, >, <, >=, <=

```
$ python3
>>> 1 == 1.0
True
>>> 2 >= 2.0
True
>>> 42.7 == Decimal('42.7')
False
```

İfade Operatörleri

- Ternary: **x** if **y** else **z**
- ||: **x** or **y**
- &&: **x** and **y**
- !: not **x**

İşlem Önceliği

Operator	Description
()	Parentheses (grouping)
f(args...)	Function call
x[index:index]	Slicing
x[index]	Subscription
x.attribute	Attribute reference
**	Exponentiation
~*x*	Bitwise not
+*x*, -*x*	Positive, negative
*, /, %	Multiplication, division, remainder
+, -	Addition, subtraction
<<, >>	Bitwise shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, <=, >, >=, <>, !=, ==	Comparisons, membership, identity
not *x*	Boolean NOT
and	Boolean AND
or	Boolean OR
lambda	Lambda expression

Bölme İşlemi

```
$ python3
>>> 5/2
2.5
>>> 5//2
2
>>> 5/2.0
2.5
>>> 5.0/2
2.5
>>> -5/2
-2.5
>>> -5//2
-3
```

```
>>> 10.0/2.0
5.0
>>> 10/2
5.0
>>> 10//2
5
```

Tam Sayı Sınırı?

```
$ python3
>>> 99999999999999999999966 + 55
100000000000000000000021

>>> 2**200
1606938044258990275541962092341162602522202993782792835301376

>>> import math
>>> math.factorial(1000)
402387260077093773543702433923003985719374864210714632543799910429938512398629020592
```

Boolean

```
>>> type(True)
<class 'bool'>
>>> isinstance(True, int)
True
>>> True == 1
True
>>> True is 1
False
>>> True or False
True
>>> True + 4
5
```

Python ve Dinamik Türler

```
>>> a = 3
```

1. *3* değerini temsil eden bir nesne yarat
2. mevcut değilse *a* isimli bir değişken yarat
3. yeni nesneyi a değişkenine bağla (referans)

- Nesne yönelimli düşünmek:
  - **Değişkenler** referans tutar.
  - **Nesneler** bellekte yer kaplar, değere sahiplerdir.
  - **Referanslar** değişkenler ile nesneler arasındaki ilişkilerdir. Yaratılır ve yıkılırlar.

Garbage Collection - Çöp Toplamak

```
>>> a = 3
>>> a = ‘selam’
>>> a = [0, 1, 1, 2, 3, 5, 8]
```

- Bir nesneye, bir değişkenden referans kalmadığı zaman *büyük ihtimalle* o nesne bellekten atılır.

Ortak Referanslar

```
$ python3
>>> a = 3
>>> b = a
>>> a = ‘selam’
>>> print(b)
3
```

```
$ python3
>>> list = [0,2,4]
>>> list2 = list
>>> list2[0] = 1
>>> print(list)
[1, 2, 4]
```

```
$ python3
>>> list = [0,2,4]
klonla:
>>> list2 = list[:]
>>> list2[0] = 1
>>> print(list)
[0, 2, 4]
```

Stringler

```
$ python3
>>> S = ‘’                          # boş string
>>> D = “başka bir string’”         # çift tırnak
>>> E = ’asd\nasd\t\x00’            # escape sequence
>>> M = ‘’’ multiline … ‘’’         # çok satırlı
>>> R = r’asd\xn’                   # raw string
>>> B = b’bytes’                    # byte string
>>> U = u’sp\u00c4m’                # unicode string
>>> print(R + U)                    # birleştirme
asd\xnspÄm
>>> print(U*3)                      # tekrarlama
spÄmspÄmspÄm
```

Stringler

```
>>>  S = ‘bu da bir string’
>>> print(S[0])
b
>>> print(S[7:12])
ir st
>>> len(S)
16
>>> “bu %s“ % S[10:16]
‘bu string’
>>> “{0} iki”.format(S[6:9])
‘bir iki’
>>> S.find(‘bir’)
6
>>> S.replace(‘bir’,’iki’)
‘bu da iki string’
```

Stringler

```
>>>  S = ‘x|y|z|asd|qwe’
>>> S.split(‘|’)
['x', 'y', 'z', 'asd', ‘qwe’]
>>> ‘@‘.join(S.split(‘|’))
‘x@y@z@asd@qwe’
```

```
>>> S.isdigit()
False
>>> S.lower()
‘x|y|z|asd|qwe’
>>> S.upper()
‘X|Y|Z|ASD|QWE’
>>> S.endswith(‘e’)
False
```

Stringler

```
>>> ‘merhaba’ “selam”
‘merhabaselam’
>>> ‘merhaba’ + ‘selam’
‘merhabaselam’
>>> str(2)+’ + ‘+str(2)+’ = %s’ % str(4)
‘2 + 2 = 4’
```

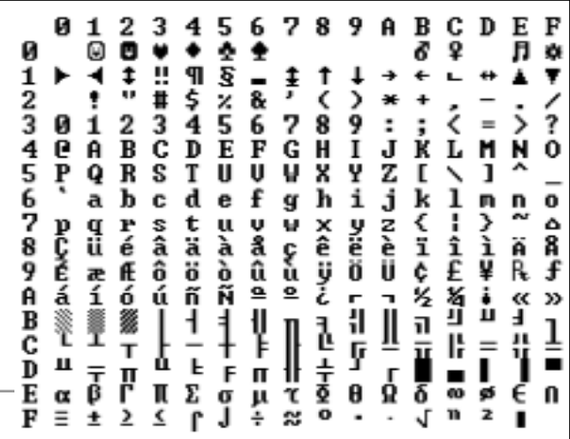
```
>>> s = ‘a\nb\tc’
>>> s
‘a\nb\tc’
>>> print(s)
a
b     c
>>> len(s)
5

>>> ‘Na’ * 8
‘NaNaNaNaNaNaNaNa’
```

Stringler

```
>>> S = 'spam'
>>> S[0], S[-2]
('s', 'a')
>>> S[1:3], S[1:], S[:-1]
('pa', 'pam', 'spa')
```

Unicode



- ASCII karakteri: 1 byte, yalnızca İngiliz alfabesi
- Unicode: evrensel standart
- Code point: **U**+0041 = A
- UTF-8: ANSI karakterler 1 byte, daha fazlası 2-6 byte
- Daha fazla bilgi: “*The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets*” - http://www.joelonsoftware.com/articles/Unicode.html

Python’da Unicode

- Python 3:
  - Tüm stringler unicode
  - 8-bit karakter dizileri byte array

```
>>> 'sp\xc4m'
'spÄm'
>>> b’a\x01c'
b'a\x01c'
>>> u'sp\u00c4m'
‘spÄm’
>>> ‘öğıçş’
‘öğıçş’
```

- Python 2:
  - Normal stringler byte array

```
>>> print u'sp\xc4m'
spÄm
>>> ‘a\x01c'
'a\x01c'
>>> b'a\x01c'
‘a\x01c'
>>> ‘öğıçş’
‘\xc3\xb6\xc4\x9f
\xc4\xb1\xc3\xa7\xc5\x9f'
```

List

- Sıralı nesne listesi
- Diğer dillerdeki array gibi, ancak daha esnek
- İç içe girebilir
- mutable — Elemanları değiştirilebilir!

```
$ python3
>>> L = []
>>> L = [1, ‘2’, 3.14]
>>> print(L[0])
1
>>> L = [‘str’, [1, 2]]
>>> L[1]
[1, 2]
>>> L[1][1]
2
```

List

```
>>> L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> L[1:5]
[1, 2, 3, 4]
>>> len(L)
10
>>> M = [10, 11]
>>> L + M
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
>>> M * 3
[10, 11, 10, 11, 10, 11]
>>> 10 in M
True
```

List

- `L.index(obj)`
- `L.count(obj)`
- `L.sort()`
- `L.reverse()`
- `L.copy() # Shallow copy`
- `L.insert(i, obj)`
- `L.clear()`
- `L.pop(i)`
- `L.remove(obj)`
- `del L[i]`
- `L[i:j] = [4, 5, 6]`

List

```
$ python3
>>> matrix = [[1,2,3], [4,5,6], [7,8,9]]
>>> matrix[1]
[4, 5, 6]
>>> matrix[2][1]
8
```

Dictionary

- Anahtar→değer çiftleri
- sırasız
- mutable — Elemanları değiştirilebilir!

```
$ python3
>>> D = {}
>>> D = {‘name’: ‘oguz’, ‘age’: 18}
>>> D[‘name’]
‘oguz’
>>> D = {‘dict’:{‘a’:’b’, ‘c’:’d’}, ‘e’:’f’}
>>> D[‘dict’]
{‘a’:’b’, ‘c’:’d’}
>>> D[‘dict’][‘a’]
‘b’
```

Dictionary

```
>>> D = dict(name='oguz', age=18)
>>> D = dict([('name','oguz'),('age',18)])
>>> list(D)
['name', 'age']
>>> D['age'] = 19
>>> D
{'age':19, 'name':'oguz'}
>>> len(D)
2
```

Dictionary

- `D.keys()`
- `D.values()`
- `D.items()`
- `D.copy()`
- `D.clear()`
- `D.update(D2)`
- `D.get(key, default?)`
- `D.pop(key, default?)`
- `del D[key]`

Tuple

- Kısaca, **immutable** List.

```
$ python3
>>> T = ()
>>> T = (0,)
>>> type(T)
<class ‘tuple'>
>>> T = (0)
>>> type(T)
<class ‘int'>

>>> T = (0, ‘asd’, 2)
>>> T = 0, ‘asd’, 2, [1, 2, 3]
>>> T
(0, 'asd', 2, [1, 2, 3])
```

Tuple ile İşleri Kolaylaştırmak

```
$ python3
>>> nudge = 1
>>> wink = 2
>>> nudge, wink = wink, nudge    # swap
>>> nudge, wink
(2, 1)

>>> [a, b, c] = (1, 2, 3)
>>> a, c
(1, 3)
>>> (a, b, c) = "ABC"
>>> a, c
('A', 'C')
```

Set (Küme)

- Immutable elemanlardan oluşan küme
- sırasız
- Her eleman bir kere yer alabilir.
- Küme işlemlerini destekliyor.

```
$ python3
>>> x = set([1, 2, ‘a’, ‘b’])
>>> x = {1, 2, ‘a’, ‘b’}
>>> x | {3, 4}                          # union
{'b', 1, 2, 'a', 4, 3}
>>> x - {‘a’, ‘b’}                      # fark
{1, 2}
```

Mutable veya Immutable

Nesne	Kategori	Mutable?
Number	Numberic	Hayır
String	Sequence	Hayır
List	Sequence	Evet
Dictionary	Mapping	Evet
Tuple	Sequence	Hayır
Set	Set	Evet

Python Shell'den Yardım Almak

```
$ python3
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__',
'__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
>>> help(list)
Help on class list in module builtins:

class list(object)
 |  list() -> new empty list
 |  list(iterable) -> new list initialized from iterable's items
 |
 |  Methods defined here:
 |
 |  __add__(...)
 |      x.__add__(y) <==> x+y
 |
 |  __contains__(...)
 |      x.__contains__(y) <==> y in x
 |
:
```

Kıyaslama

```
$ python3
>>> L1 = [1, ('a', 3)]
>>> L2 = [1, (‘a’, 3)]
>>> L1 == L2, L1 is L2
(True, False)

>>> S1 = ‘lol’
>>> S2 = ‘lol’
>>> S1 == S2, S1 is S2
(True, True)

>>> S3 = ‘daha uzun bir string’
>>> S4 = ‘daha uzun bir string’
>>> S3 == S4, S3 is S4
(True, False)
```

Yorum Satırları

- Başında **#** olan her satır, Python yorumlayıcısı tarafından ihmal edilir.
- Çok satırlı yorum? **Aslında yok**.
- Çözüm olarak çok satırlı string üretmek:

```
''' bu string
    bir değişkene atanmadıkça
    programa bir etkisi yoktur.
'''
```

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`

Kod Bloğu

C, C++, Objective-C, Java, JavaScript … :

```
if ( x > y ) {
   x = 1;
   y = 2;
}
```

Python:

```
if ( x > y ):
   x = 1
   y = 2
```

Özel durumlar

```
$ python3
```

Hani noktalı virgül yoktu?

```
>>> a=4;b=2;c=m()
```

Aynı satırda:

```
>>> if a == 4: print(‘yes’)
```

Birden fazla satırda:

```
>>> if( a == 4 and
        b == 2 ):
      print(‘yes’)
```

İnteraktif Programlar

```
$ python3
>>> a = input(‘bir sayı girin: ’)
bir sayı girin: 4
>>> a
4
>>> type(a)
<class 'str'>
```

If / elif / else

```
# python3
a = input(‘bir tamsayı girin: ’) # str
a = int(a)

if a > 0:
  print(‘pozitif bir sayı girdiniz’)
elif a < 0:
  print(‘negatif bir sayı girdiniz’)
else:
  print(‘sıfır girdiniz’)
```

while

```
$ python3
>>> i = 0
>>> while i < 10 :
...     print(i)
...     i = i + 1
...
0
1
2
3
4
5
6
7
8
9
```

while / else

```
$ python3
>>> i = 0
>>> while i > 0 :
...         print(i)
...         i = i + 1
... else :
...     print(‘bu döngü çalışmadı’)
...
bu döngü çalışmadı
```

pass, break, continue

---

pass: boşluk doldurma

```
if( itemCount == 0 ):
   # TODO: implement this condition
   pass
```

break: döngüden çıkma

```
while ( i > 0 )
   if( objects[i] == None ):
      break
```

continue: bir sonraki adıma geçme

```
while ( i > 0 )
   if( objects[i] == None ):
      continue
```

Ternary

```
$ python3
>>> x = 0
>>> y = -1 if x >= 0 else 1
>>> y
-1
```

Alıştırma 1: Fibonacci

- List nesnesini kullanarak, Fibonacci sayılarını hesaplayalım.

- Hesapladığınız sayıları bir List'te saklayalım.

- [0, 1, 1, 2, 3, 5, 8, 13 ...]

Alıştırma 2: Palindrom

- Verilen bir string'in palindrom olup olmadığına karar veren bir program yazalım.

- Palindrom: baştan ve sondan aynı şekilde okunabilen kelime

- Örnek: abba, ey edip adanada pide ye

Alıştırma 3: Farklı harfler

- Set kullanarak, verilen iki string'de ortak olan ve olmayan karakterleri bulalım.

- Örnek: araba, apartman

- yalnızca ilkinde bulunan harfler: b

- yalnızca ikincide bulunan harfler m, t, n, p

- her iki stringde de bulunan harfler: r, a

Teşekkürler!

Bir sonraki oturumda görüşmek üzere.