Bileşik veri tipi

→ şimdiye kadar gördüğümüz tipler: int, float, bool, NoneType, str

→ str diğerlerinden farklı alt parçalardan oluşuyor, char

→ daha küçük yapılardan oluşan tiplere **bileşik veri türleri** denir

→ bütün olarak veya parçalarıyla çalışabiliriz

→ köşeli- [] dizgideki karakterlere eriştirir (PB:5354)

```
>>> fruit = "banana"
>>> letter = fruit[1]
>>> print letter
a
```

→ indisler, sıfırdan başlar

Uzunluk

→ dizgi uzunluğunu öğrenmek (PB:5357)

```
>>> fruit = "banana"
>>> len(fruit)
6
```

→ dizginin son karakterine ulaşmak için

```
>>> fruit[len(fruit)]
Traceback (most recent call last):
  File "<input>", line 1, in <module>
IndexError: string index out of range
>>> fruit[len(fruit) - 1]
'a'
```

→ indisler sıfırdan başlıyordu

→ negatif indisler dizginin sonundan başlandığını gösterir

→ dizginin son elemanı, dizgi[len-1]== dizgi[-1]

Gezinme ve for döngüsü

→ genelde dizginin karakterlerini işlememiz gerekir

→ döngü kurmak gerekecek, while (PB:5359)

```
>>> fruit = "banana"
>>> ind = 0
>>> while ind < len(fruit):
...     ch = fruit[ind]
...     print ch
...     ind += 1
...
...
b
a
n
a
n
a
```

for döngüsü

→ daha kolay söz dizimi sağlar (PB:5360)

```
>>> fruit = "banana"
>>> for ch in fruit:
...     print ch
...
...
b
a
n
a
n
a
```

ördek yavruları

→ Robert McCloskey (PB:5361)

```
>>> prefixes = "JKLMNOPQ"
>>> suffix = "ack"
>>> for letter in prefixes:
...     print letter + suffix
...
...
Jack
Kack
Lack
Mack
Nack
Oack
Pack
Qack
```

Karakter dizisi dilimleri

→ dizginin altparçasına **dilim** (substring)

→ dilim seçmek, karakter seçmeye benzer (PB:5426)

```
>>> str = "OMU, Muhendislik, Bilgisayar"
>>> print str[0:3]
OMU
>>> print str[5:16]
Muhendislik
>>> print str[18:28]
Bilgisayar
```

→ dilim= str[n:m], n. karakterle m. karakter arasındaki altdizgi=dilim

→ n içerilir, m içerilmez

→ eğer başlangıcı verilmezse 0, bitişi verilmezse dizgi uzunluğu alınır

```
>>> print str[ :5]
OMU,
>>> print str[5: ]
Muhendislik, Bilgisayar
>>> str[:]
'OMU, Muhendislik, Bilgisayar'
```

Karakter dizisi karşılaştırma

→ karşılaştırma işleci kullanılabilir (PB:5427)

```
>>> parola = raw_input("Parolayi giriniz... ")
Parolayi giriniz... abc123
>>> if parola == "abc123":
...     print "Basarili giris yaptiniz!"
...
...
Basarili giris yaptiniz !
```

→ sıralama yapılarında kullanabilirsiniz

```
if word < "muz":
    print "Kelimeniz," + word + ", muzdan once gelir."
elif word > "muz":
    print "Kelimeniz," + word + ", muzdan sonra gelir."
else:
    print "Evet, hic muzumuz yok!"
```

→ ASCII tablosuna göre!

Karakter dizileri değişmez

→ dizgi **değişmez** (immutable) nesnelerdir (PB:5428)

```
>>> greeting = "Merhaba, dunya!"
>>> greeting[0] = 'N'                # ERROR!
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> print greeting
Merhaba, dunya !
```

→ verilen hata mesajına dikkat!

→ çözüm (PB:5429)

```
>>> greeting = "Merhaba, dunya!"
>>> newGreeting = 'N' + greeting[1:]
>>> print newGreeting
Nerhaba, dunya !
```

in işleci

→ in: altdizginin, dizgi içerisinde var olup olmadığını sınamada

```
>>> 'p' in 'apple'
True
>>> 'i' in 'apple'
False
>>> 'ap' in 'apple'
True
>>> 'pa' in 'apple'
False
```

örnek

→ tüm sesli harfleri uzaklaştırmak

```
1       def remove_vowels(s):
2           """\
3               Tum sesli harfleri dizgiden cikarir
4               >>> str = "merhaba, dunya!"
5               'mrhb, dny!'
6           """
7
8           vowels = "aeiouAEIOU"
9           s_without_vowels = ""
10          for letter in s:
11              if letter not in vowels:
12                  s_without_vowels += letter
13          return s_without_vowels
```

Bir bulma(find) fonksiyonu

→ dizgideki karakteri bulduğu yer

```
1       def find(strng, ch):
2           index = 0
3           while index < len(strng):
4               if strng[index] == ch:
5                   return index
6               index += 1
7           return -1
```

→ döngü içerisinde return

→ bulamazsan -1

→ sıra sizde: d07_strfind, d07_strcmp, , d07_islower, d07_upper, d07_strupper,...

Döngü ve sayma

→ harf dizgi içerisinde kaç kez geçiyor

```
1       def strcnt(str, ch):
2           count = 0
3           for x in str:
4               if x == ch:
5                   count += 1
6
7           return count
```

→ sıra sizde: d07_hist: alfabedeki tüm harflerin sıklığı

İsteğe bağlı parametreler

→ isteğe bağlı (optional) parametre kullanımı

```
1       def find2(strng, ch, start=0, step=1):
2           index = 0
3           while index < len(strng):
4               if strng[index] == ch:
5                   return index
6               index += step
7           return -1
```

→ sıra sizde: d04_kur geliştirin

string modülü

- → dizgiler için faydalı işlevler: import string
- → sağladığı olanaklar: dir(string)
- → bu modüldeki bir işlev için yardım: help(string.find)

Karakter sınıflandırma

→ karakter: büyük, küçük harf, sayı mı? sınıflandır

```
1  def is_lower(ch):
2      return ch in string.lowercase
```

→ alternatif olarak

```
1  def is_lower(ch):
2      return 'a' <= ch <= 'z'
```

Boş karakterler (whitespace)

→ boş karakterler: boşluk, tab (t) ve yeni satır (n)

→ string.whitespace

Karakter dizisi biçimlendirme

→ Karakter dizisi biçimlendirme işleminin sözdizimi şu şekildedir

→ "<BICIM>" % (<DEGERLER>)

→ ekrana çıktı verirken kullanmıştık (PB:5430)

```
>>> "His name is %s."  % "Arthur"
'His name is Arthur.'
>>> name = "Alice"
>>> age = 10
>>> "I am %s and I am %d years old." % (name, age)
'I am Alice and I am 10 years old.'
>>> n1, n2 = 4, 5
>>> "2**10 = %d and %d * %d = %f" % (2**10, n1, n2, n1 * n2)
'2**10 = 1024 and 4 * 5 = 20.000000'
```

→ daha fazla bilgi: http://docs.python.org/lib/typesseq-strings.html

tablolar

→ örnek tablo yapımı (PB:5432)

```
>>> i = 1
>>> while i <= 10:
...     print "%-4d%-5d%-6d%-8d%-13d%-15d" % \
...         (i, i**2, i**3, i**5, i**10, i**20)
...     i += 1
...
...
1   1    1     1       1             1
2   4    8     32      1024          1048576
3   9    27    243     59049         3486784401
4   16   64    1024    1048576       1099511627776
```

→ -: sola yaslama

→ 13d: en az 13 karakterlik yer

PIL: Python Imaging Library

→ resim işleme için kullanabilirsiniz

→ PB:5434

```
1   import Image
2
3   im = Image.open("GvR.jpg")
4   W, H = im.size           # genislik, yukseklik
5   r, g, b = im.split()     # RGB split
6   # point erisimi
7   pr, pg, pb = r.load(), g.load(),  b.load()
8
9   # isleme
10  for x in range(W):
11      for y in range(H):
12          if pr[x, y] < 128:
13              pr[x, y] = 0
14          else:
15              pr[x, y] = 255
16
17          pg[x, y] = 255 - pg[x, y]
18          pb[x, y] = pb[x, y] * 1.2
19
20  im2 = Image.merge("RGB", (r, g, b))
21  im2.rotate(45).show()
```

sıra sizde

- → d07_strfind, d07_strcmp, , d07_islower, d07_upper, d07_strupper
- → d07_hist: alfabedeki tüm harflerin sıklığı
- → d04_kur geliştirin
- → d07_count_letters(str, ch): str dizgisinde ch'ların sayısını döndür

d07_stringtools.py:

- → reverse(str),
- → mirror(str),
- → remove_letter(letter, strng),
- → is_palindrome(s)
- → count(sub, s)
- → remove(sub, s)
- → remove_all(sub, s)