Python ile Programlamaya Giriş

DERS 8: DEMETLER VE SÖZLÜKLER

DR. HÜSEYİN BAHTİYAR

Toplu veri "collection" ne değildir

- Bir çok değişkende değişken bir değerlidir. Değişkene yeni atandığında eski değerin üzerine yazılır.

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

Collection gibidir

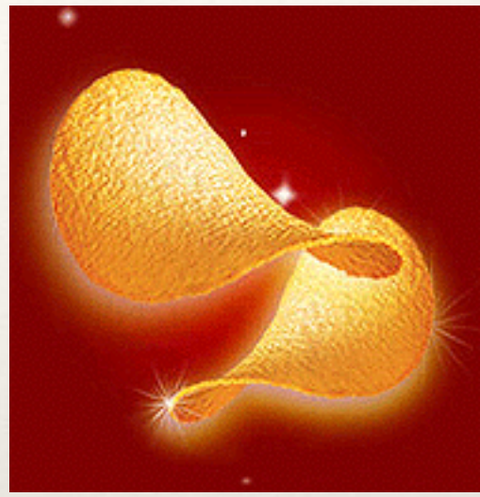
- collection tek bir değişkene birden fazla değer atamaya izin verir.

- collection kullanışlıdır çünkü bir çok değeri tek bir paket ile istediğimiz yere taşıyabiliriz.

Collection karşılaştırması

- Liste

Lineer veri topluluğu, değişkenler benzer sıralamadadır



- Sözlük

Her bir değişkenin kendine özel etiketi olan bir çanta gibi düşünebiliriz

Sözlükler (Dictionaries)

- Sözlükler Pythondaki en güçlü veri yapısıdır.
- Veri tabanı gibi işlemleri, hızlıca yapmamızı sağlarlar.
- Farklı programlama dillerinde farklı şekillerde isimlendirilmişlerdir.

  - Associative Arrays - Perl / PHP
  - Properties veya Map veya HashMap - Java
  - Property Bag - C# / .Net

Sözlükler (Dictionaries)

- Liste indisleme yaparken listede bulunduğu yere göre indis atar
- Sözlükler ise hiç bir sıralama olmayan veri çantaları gibidir.
- indisleme işleminde programcı sözlüğe her bir değişkene özel "etiket" verir.

```
>> purse = dict()
>> purse['money'] = 12
>> purse['candy'] = 3
>> purse['tissues'] = 75
>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 3}
>> print(purse['candy'])
3
>> purse['candy'] = purse['candy'] + 2
>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 5}
```

Liste ile Sözlüğü karşılaştırmak

- Sözlükler de liste gibilerdir ancak listedeki gibi liste sırası yerine, her bir değişkeni için için özel anahtar değişkenler kullanıyoruz.

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print(ddd)
{'course': 182, 'age': 23}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

Liste

Anahtar	Değişken
[0]	21
[1]	183

lst

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print(ddd)
{'course': 182, 'age': 23}
```

Sözlük

Anahtar	Değişken
['course']	182
['age']	21

ddd

Sözlük Yaratmak

- Sözlük yaratırken süslü parantez kullanırız { } bu süslü parantezin arasına anahtar_sözcük : değişken girdi yaparız.
- Eğer boş bir sözlük oluşturmak istersen süslü parantez açıp kapamamız yeterlidir { }

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(jjj)
{'jan': 100, 'chuck': 1, 'fred': 42}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```

Örnek en sık isim?

marquard

zhen

csev

cwen

marquard

zhen

csev

cwen

zhen

marquard

csev

zhen

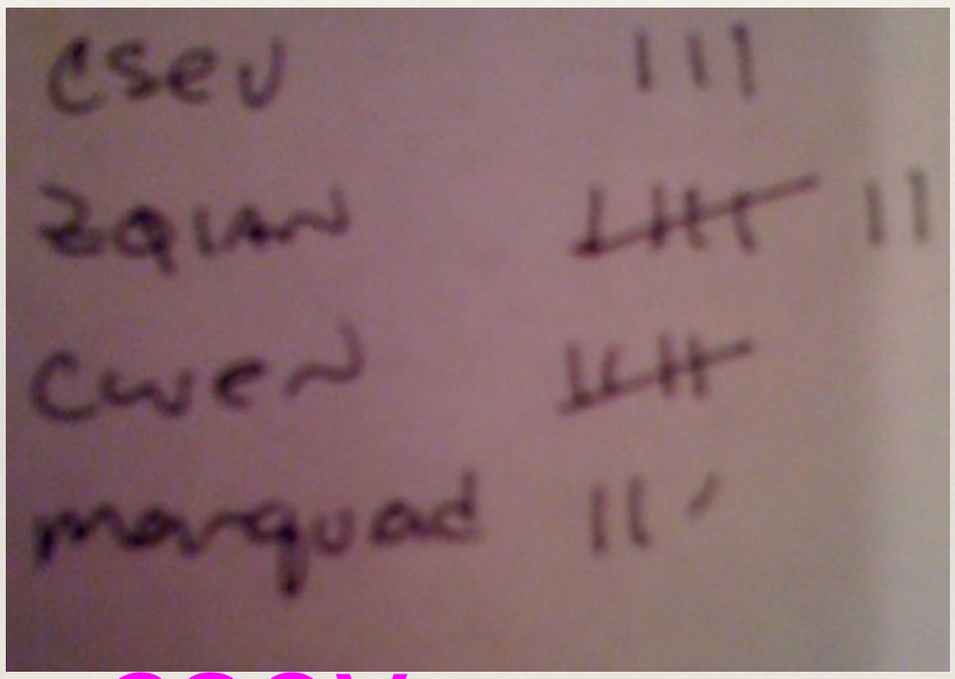
Örnek en sık isim?

marquard

zhen

csev

zhen



csev

cwen

cwen

zhen

csev

marquard

zhen

Örnek en sık isim?

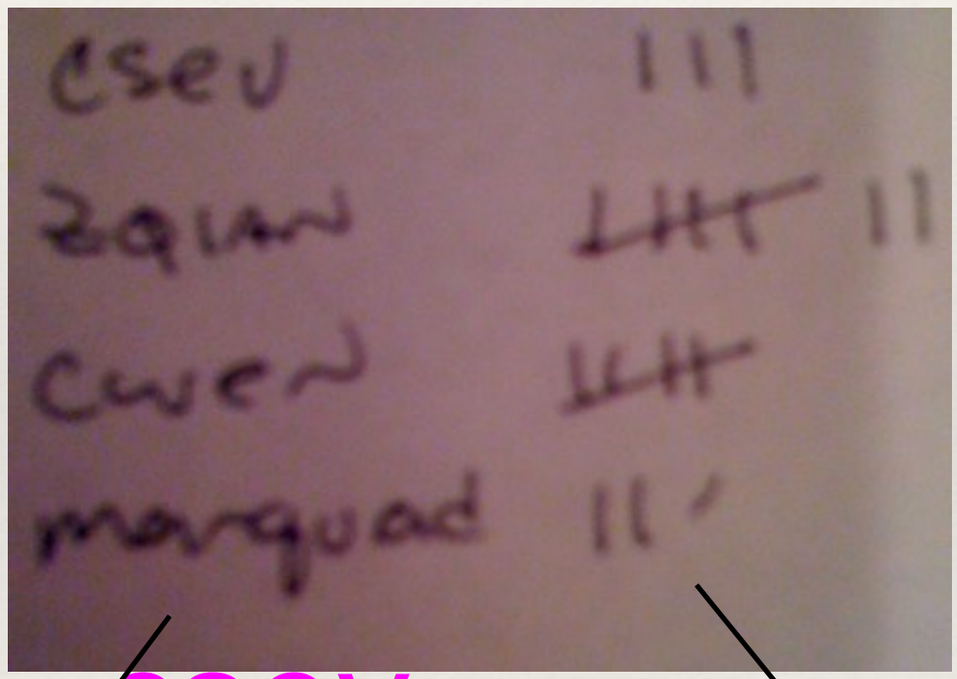
marquard

zhen

csev

zhen

cwen



csev

cwen

zhen

csev

marquard

zhen

Anahtar

Değişken

Örnek en sık isim?

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
   if name not in counts:
      counts[name] = 1
   else :
      counts[name] = counts[name] + 1
print(counts)
```

Örnek en sık isim?
Daha kolayı :)

get() fonksiyonu ile ilk değeri 0 olan ve değişken ile her karşılaştığında 1 arttıran komut yazabiliriz.

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

Default

{'csev': 2, 'zqian': 1, 'cwen': 2}

Örnek Saymak

```
counts = dict()
line = input('Enter a line of text:')
words = line.split()

print('Words:', words)
print('Counting...')

for word in words:
    counts[word] = counts.get(word,0) + 1
print('Counts', counts)
```

Örnek Saymak

```
counts = dict()
line = input('Bir yazi girin:')
words = line.split()

print('kelimeler:', words)
print('sayiliyor...')

for word in words:
    counts[word] = counts.get(w
print('sonuc', counts)
```

```
python wordcount.py
Bir yazi girin:
the clown ran after the car and the car ran into the tent and
the tent fell down on the clown and the car

Kelimeler: ['the', 'clown', 'ran', 'after', 'the', 'car', 'and',
'the', 'car', 'ran', 'into', 'the', 'tent', 'and', 'the', 'tent',
'fell', 'down', 'on', 'the', 'clown', 'and', 'the', 'car']
Sayiliyor...

sonuc {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into': 1, 'after':
1, 'clown': 2, 'down': 1, 'fell': 1, 'the': 7, 'tent': 2}
```

Demetler (Tuples)

- Demetler (Tuples) de Liste tipine benzeyen bir veri yapısıdır ilk elemanı 0 dan başlar.

```
>>> x = ('Glenn', 'Sally', 'Joseph')
>>> print(x[2])
Joseph
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for iter in y:
...     print(iter)
...
1
9
2
>>>
```

Fakat Demetler Değiştirilemez

- Demeti tanımlarken Listenin aksine [ ] değil, normal parantez içerisinde tanımlarız ( )
- Listelerde öğrendiğimizin aksine demetlerin yapısını değiştiremeyiz (birşey ekleyip çıkaramayız)

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>>[9, 8, 6]
>>>
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback:'tuple' object
does
not support item
Assignment
>>>
```

Demet ile Yapamayacaklarımız

```
>>> x = (3, 2, 1)
>>> x.sort()
Traceback:
AttributeError: 'tuple' object has no attribute 'sort'
>>> x.append(5)
Traceback:
AttributeError: 'tuple' object has no attribute 'append'
>>> x.reverse()
Traceback:
AttributeError: 'tuple' object has no attribute 'reverse'
>>>
```

Komutlar?

```
>>> l = list()
>>> dir(l)
['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']

>>> t = tuple()
>>> dir(t)
['count', 'index']
```

Peki niye Demet?

- Demetler değiştirilemez olduğundan ötürü hafıza kullanımı ve performans açısından daha basit ve verimlidir.
- Bu sebeple programımızda “geçici değişkenler” kullanmak istediğimizde listeler yerine demetleri kullanmamız performans açısından daha verimli olur.

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

… Insert new Contributors and Translators here