Python Programlama

Ders 6

Ali Mertcan KOSE Msc.
amertcankose@ticaret.edu.tr

İstanbul Ticaret Üniversitesi

İSTANBUL TİCARET
ÜNİVERSİTESİ

Nesne Türleri

Skaler nesneler: erişilebilir bir iç yapıya sahip değildir

- Pythonda Skaler nesneler: int, float

Yapılandırılmış nesneler:

- Veri yapıları, bazı verileri bir arada tutabilen yapılardır. İlgili verilerin bir koleksiyonunu depolamak için kullanılırlar.
- Python 3.7'deki yerleşik veri yapıları str, list, tuple, range ve dictionary'yi içerir.

Yapılandırılmış Nesneler: str

Dizeler, bir dizeden tek tek karakterleri çıkarmak için dizin oluşturmayı ve alt dizeleri çıkarmak için dilimlemeyi kullanabileceğiniz için yapılandırılmıştır.

Daha önce tartışıldığı gibi, dize nesneleri yerleşik işlevselliği oluşturmaktadır.

Dize nesneleri sabittir , yani dize oluşturulduktan sonra değiştirilemez.

Bir dizeyi değiştirmek her zaman yeni bir dize oluşturulmasına neden olur.

Yapılandırılmış Nesneler: tuples

Tuples, eleman tiplerinin bir karışımını içerebilen sıralı bir eleman dizisidir.

Tuples değişmez, öğe değerlerini değiştiremezsiniz.

Tuples parantez () kullanılarak temsil edilir.

tuples Örneği

```
#create an empty tuple
```

t1 = ()

```
#create a tuple containing 3 values
```

t2 = (1,“Two”, 3)

```
#display the tuples
```

print(t1)

print(t2)

tuples Örneği

```
#display an element in a tuple
```

```
print( t2[1] )
```

```
#display the type of the element
```

```
print( type(t2[1]))
```

```
#tuples are immutable
```

```
#t2[0] = 5--> TypeError: 'tuple' object does not support i
```

tuples Örneği

Output:

()

(1, ‘Two’, 3)

Two

<class ‘str’>

Yapılandırılmış Nesneler: tuples

Dizeler gibi, tuples'lar birleştirilebilir, endekslenebilir, dilimlenebilir ve tekrarlanabilir.

```
#concatenating tuples
```

t1 = ('a', 'b', 5)

t2 = (7,)

t1 = t1 + t2

print(t1)

```
#indexing with tuples
```

print(t1[2])

Yapılandırılmış Nesneler: tuples

```
#slicing tuples
```

print(t1[1:3])

```
#repeating tuples
```

t3 = 2 * t1

print(t3)

```
#nesting tuples
```

t4 = ((1,'z'), 8, ('hi', 2, 'u'))

print(t4)

Yapılandırılmış Nesneler: tuples

Output:

('a', 'b', 5, 7)

5

('b', 5)

('a', 'b', 5, 7, 'a', 'b', 5, 7)

((1, 'z'), 8, ('hi', 2, 'u'))

tuples Örneği

te = ()

t= (2,“mit”, 3)

t[0]

(2,“mit”,3) + (5,6)

t[1:2] –> slice tuple, evaluated to (“mit”,)

[1:3] –> slicetuple, evaluated to (“mit”, 3)

len(t) –> evauates to 3

t[1] = 4 –> gives error, can’t modify object

tuples Örneği

t1 = (1, ‘two’, 3)

t2 = (t1, 3.25)

print(t2)

print(t1 + t2)

print((t1+t2)[3])

print((t1+t2)[2:5])

Output:

((1, ‘two’, 3), 3.25)

(1, ‘two’, 3, (1, ‘two’, 3), 3.25)

(1, ‘two’, 3)

(3, (1, ‘two’, 3), 3.25)

tuples Örneği

- Değişken değerleri değiştirmek için uygun şekilde kullanılır.

1. örnek (yanlış)

x=y

y=x

2. örnek (doğru)

temp = x

x=y

y=temp

tuples Örneği

3. örnek (doğru)

(x,y) = (y,x)

- Bir işlevden birden fazla değer döndürmek için kullanılır.

```
def quotient_and_remainder(x,y):
q= x//y
r= x%y
return (q,r)
(quot, rem) = quotient_and_remainder(4,5)
```

Bir tuple Çaprazlaması

Bir tuple elemanlarının toplamını hesaplayın.

Ortak desen, tuple elemanları üzerinde yineleyin.

total = 0

for i in range (len(T)):

total+=T[i]

print total

Bir tuple Çaprazlaması

total = 0

for i in range (T):

total+=i

print total

Not:

- Tuple elemanları 0 ile len(T)-1
- range(n) 0'dan n-1 kadar gider.

Yapılandırılmış Nesneler: ranges

Dizeler ve Tuples gibi, ranges değişmez.

Bir değer aralığı döndürmek için range() fonksiyonunu kullanabiliriz ve fonksiyon 3 parametre alır (start, stop, step)

Tuples üzerindeki tüm işlemler, birleştirme ve yineleme dışında aralıklarla da kullanılabilir.

ranges Örneği

```
#create a range

r1 = range(1,11,2)

for i in r1:

print(i, end=" ")

print()

#create a range omit step

for j in range(2,12):

print(j, end=" ")

print()
```

ranges Örneği

```
#create a range omit start and step
```

for k in range(10):

print(k,end=" ")

print()

```
#slicing/indexing ranges
```

print(range(10)[2:4][1])

Output:

1 3 5 7 9

2 3 4 5 6 7 8 9 10 11

0 1 2 3 4 5 6 7 8 9

3

ranges Karşılaştırma

Eşitlik operatörü (==) range nesnelerini karşılaştırmak için kullanılabilir.

İki aralık aynı tamsayı dizisini temsil ediyorsa Doğru, temsil etmiyorsa Yanlış döndürür.

Aralıkları karşılaştırırken, değerlerin ve sıralarının eşit olması için aynı olması gerekir.

Örnek:

range(0,7,2) = = range (0,8,2) -> evaluates to true

range(0,7,2) == range(6,-1,-2) -> evaluated to False

Yapılandırılmış Nesneler: listeler

Liste, ilgili değerlerin bir dizisini depolamak için kullanılır.

Listeler, dizin tarafından erişilebilen sıralı bir bilgi dizisidir.

Bir liste köşeli parantezlerle gösterilir, [ ]

Bir liste genellikle homojen elementler içerir.

Liste öğeleri değiştirilebilir, böylece bir liste değiştirilebilir.

listeler

```
#creating an empty list
```

a_list = []

```
#creating a list and initializing values
```

L = [2,8,3,6]

```
#creating a list and initializing values
```

print(L)

```
#display the number of elements in a list
```

print(len(L))

print(L)

listeler

```
#indexing from zero-accessing an element
```

print(L[0])

print(L[2]+1)

print(L[3])

```
#print(L[4])--> no element at index 4
```

```
#indexing with variables
```

i = 2

L[i-1]

listeler

```
#updating an element
```

L[1]= 12

print(L)

Output:

[2, 8, 3, 6]

4

2

4

6

[2, 12, 3, 6]

Liste ve Değişkenlik

Listeler değişebilir!

L = [2, 1, 3]

L[1] = 5

L şimdi [2, 5, 3], bunun aynı nesne olduğuna dikkat edin

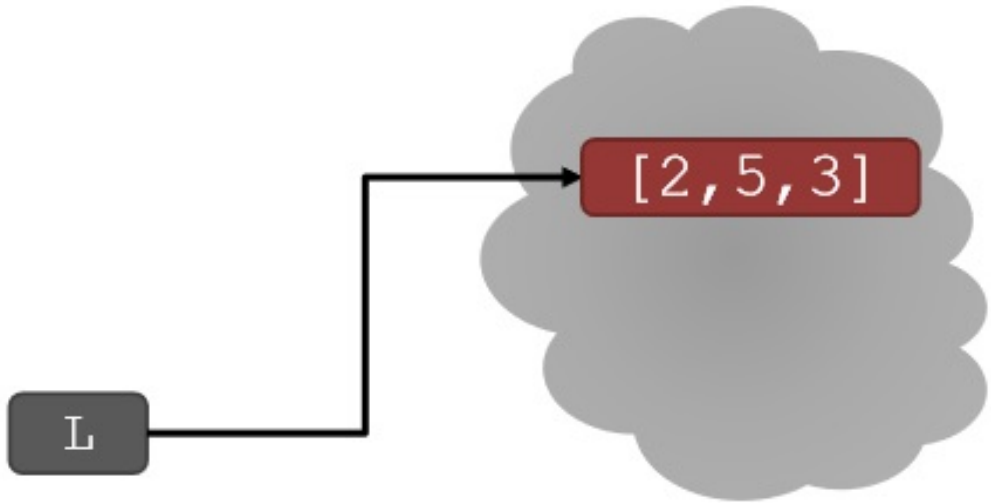

Figure 1: Liste ve Değişkenlik

Listede Geçiş yapma

Bir listenin öğelerinin toplamını hesaplayın.

Ortak desen, liste öğeleri üzerinde yineleyin.

```
total = 0
for i in range(len(L)):
    total += L[i]
print total
```

```
total = 0
for i in L:
    total += i
print total
```

Figure 2: Liste ve Değişkenlik

Note:

Liste öğeleri 0 ila len (L)-1 kadar indekslenir
range(n) 0'dan n 1'e gider

Liste İşlemleri İliştirmek

Bir listenin sonuna ekleme işleviyle öğeler ekleyebiliriz

L.append(e) –> adds the element,e, to the end of L

Listeyi değiştirir!

L = [2,1,3]

L.append(5)–> L is now [2,1,3,5]

Liste İşlemleri Ekleme

Listede belirli bir konuma eleman ekleyebiliriz.

L.insert(i, e) –> inserts the element,e, into L at index i

Örnek

friends = [‘Harry’, ‘Emily’, ‘Bob’, ‘Jane’]

friends.insert(1,‘Cindy’)

print(friends)

Output:

[‘Harry’, ‘Cindy’, ‘Emily’, ‘Bob’, ‘Jane’]

Liste İşlemleri-Bir Öğeyi Bulma

Bir elementin oluştuğu konumu bulabiliriz.

L.index(e)

L'de e'nin ilk geçtiği yerin indeksini verir, e L'de değilse çalışma zamanı hatası verir.

Örnek

friends = ['Harry', 'Emily', 'Bob', 'Jane', 'Emily']

n = friends.index('Emily')

print(n)

n = friends.index('Emily', n+1)

print(n)

Liste İşlemleri-Bir Öğeyi Bulma

Output:

1

4

Liste İşlemleri-Bir Öğeyi Bulma

Öğe listede yoksa index hataya neden olacağından, index işlevini çağırmadan önce in operatörüyle test etmek genellikle iyi bir fikirdir.

Örnek

friends = [‘Harry’, ‘Emily’, ‘Bob’, ‘Jane‘, ‘Emily’]

if ‘Emily’ in friends:

n = friends.index(’Emily‘)

else:

n = None

Liste İşlemleri-Bir Öğeyi Bulma

Özel durumu işleyen alternatif çözüm.

Örnek

friends = [‘Harry’, ‘Emily’, ‘Bob’, ‘Jane‘, ‘Emily’]

try:

n = friends.index(’Emily‘)

except:

n = None

Liste İşlemleri-Bir Öğeyi Kaldırma

Öğeleri bir listeden dizine göre kaldırabiliriz

L.pop(i)

–> removes and returns the item at index i in L,

–> if no index is specified it removes the last element

–> if i is not a valid index, a runtime error will occur

Bir elemanı değerine göre de kaldırabiliriz

L.remove(e)

–> deletes the first occurrence of e from

–> if e does not exist in the list, a runtime ValueError will occur.

Liste İşlemleri-Bir Öğeyi Kaldırma

Örnek:

friends = [‘Harry’, ‘Emily’, ‘Bob’, ‘Jane’]

friends.pop(1)

print(friends)

Liste İşlemleri-Bir Öğeyi Kaldırma

Output:

['Harry', 'Bob', 'Jane']

Örnek:

friends = ['Harry', 'Emily', 'Bob', 'Jane']

friends.pop()

print(friends)

Output ['Harry', 'Emily', 'Bob']

Örnek:

friends = ['Harry', 'Emily', 'Bob', 'Jane']

friends.remove('Bob')

Output

['Harry', 'Emily', 'Jane']

Liste İşlemleri- Birleştirme

İki listenin birleştirilmesi, ilk listenin öğelerini ve ardından ikinci listenin öğelerini içeren yeni bir listedir. İki listeyi birleştirdiğimizde hiçbir yan etkisi yoktur, yani yeni bir liste oluşturulur ve birleştirilen orijinal listeler mutasyona uğramaz.

Örnek

myFriends = [‘Jane’, ‘Bob’, ‘Emily’]

yourFriends = [‘Cindy’, ‘John’]

ourFriends = myFriends + yourFriends

Output

[‘Jane’, ‘Bob’, ‘Emily‘, ‘Cindy’, ‘John’]

Aşağıdaki örneği izleyin

```
L1 = [1,2,3]
L2 = [4,5,6]
L3 = L1 + L2
print('L3 =',L3)
L1.extend(L2)
print('L1 =',L1)
L1.append(L2)
print('L1 =',L1)
```

Aşağıdaki örneği izleyin

```
L1 = [1,2,3]
L2 = [4,5,6]
L3 = L1 + L2
print('L3 =',L3)
L1.extend(L2)
print('L1 =',L1)
L1.append(L2)
print('L1 =',L1)
```

Aşağıdaki örneği izleyin

```
L1 = [1,2,3]
L2 = [4,5,6]
L3 = L1 + L2
print('L3 =',L3)
L1.extend(L2)
print('L1 =',L1)
L1.append(L2)
print('L1 =',L1)
```

Aşağıdaki örneği izleyin

Output

L3 = [1, 2, 3, 4, 5, 6]

L1 = [1, 2, 3, 4, 5, 6]

L1 = [1, 2, 3, 4, 5, 6, [4, 5, 6]]

Liste İşlemleri-Eşitlik Testi

Eşitlik operatörü (==), iki listenin aynı öğelere aynı sırada sahip olup olmadığını karşılaştırmak için kullanılabilir.

Örnek:

[1,4,9] == [1,4,9] is

True , but

[1,4,9] == [4,1,9] is

False

The opposite of == is !=.

Örnek:

[1,4,9] != [4,9] is

True

Liste İşlemleri-toplamı, maks, min

Bir listenin toplamını, maksimum öğesini, minimum öğesini bulmak istediğinizde toplam, maksimum, min, fonksiyonlarını kullanabilirsiniz.

Örnek:

x=[1,16,9,4]

sum(x)

will give 30

max(x)

will give 16

min(x)

will give 1

Liste İşlemleri-toplamı, maks, min

x.sort()

will make x=[1,4,9,16]

x.sort(reverse=True)

will make x=[16,9,4,1]

Liste İşlemleri-Sıralama

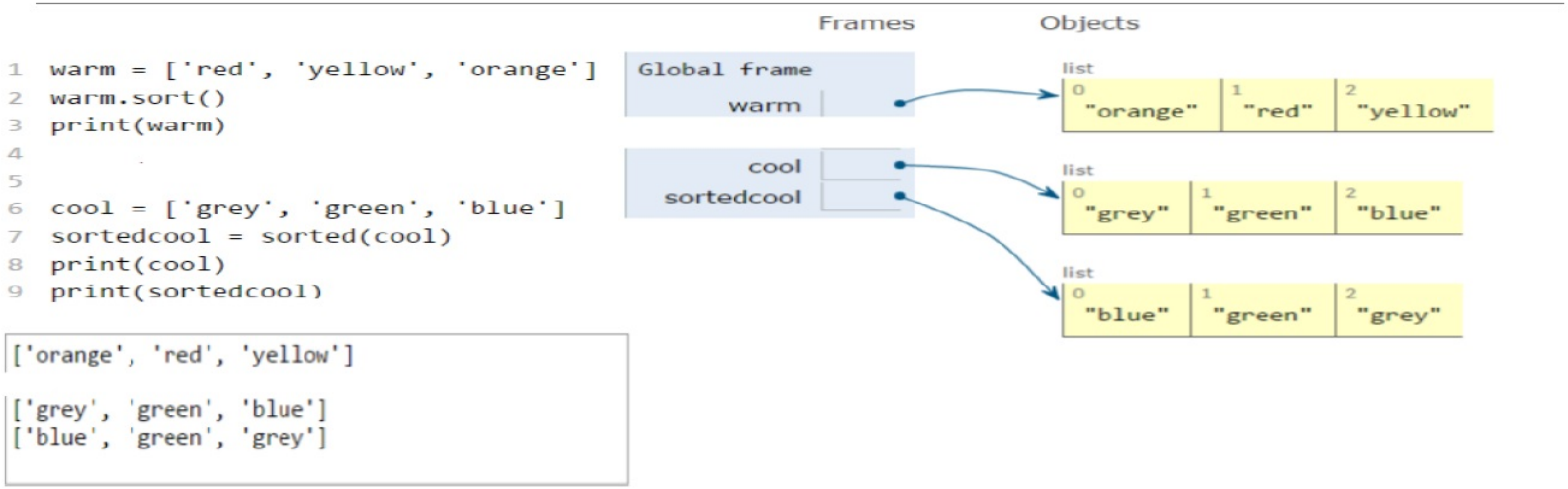

Figure 3: Liste İşlemleri Sıralama

Liste İşlemleri-Sıralama

sort(): öğesinin çağrılması listeyi değiştirir, hiçbir şey döndürmez

İki sürüm: x.sort() x=[1,4,9,16] yapar.

x.sort(ters doğru) x=[16,9,4,1] yapar.

sorted() çağrısı listeyi değiştirmez, bir değişkene sonuç ataması gerekir.

Liste İşlemleri-Özet

Function	Purpose
len(L)	L içindeki öğelerin sayısını verir.
L.append(e)	e nesnesini L nesnesinin sonuna ekler.
L.count(e)	e'nin L içinde kaç kez geçtiğini verir.
L.insert(i,e)	e nesnesini i dizininde L'ye ekler.
L.extend(L1)	L1 listesindeki öğeleri L'nin sonuna ekler.
L.remove(e)	e'nin ilk geçtiği yeri L'den siler
L.index(e)	L'de e'nin ilk oluşumunun dizinini verir ve e L'de değilse bir çalışma zamanı hatası verir.
L.pop(i)	L'deki i dizinindeki öğeyi kaldırır ve döndürür ve L boşsa veya dizin listenin sınırlarının dışındaysa bir çalışma zamanı hatası verir. i atlanırsa, son öğeyi döndürür (-1 dizinindeki öğe)
L.reverse()	L'deki öğelerin sırasını tersine çevirir.

Dizeleri Listeler Olarak Bölme

s.split(d) – sınırlayıcı olarak d kullanarak s'yi böler. s alt dizelerinin listesini döndürür. d atlanırsa, alt dizeler rastgele boşluk karakterleri dizesiyle ayrılır.

s = 'dog,cat,mouse,horse'

words = s.split(',')

print(words)

Dizeleri Listeler Olarak Bölme

Output:

['dog','cat','mouse','horse']

words2 ='dog cat mouse horse'.split()

print(words2)

Output:

['dog','cat','mouse','horse']

Sözlük

Sözlük, anahtarlar ve değerler arasındaki ilişkileri depolayan bir kapsayıcıdır. Benzersiz anahtarları değerlerle eşlediği için harita olarak da bilinir.

Her anahtar bir değerle ilişkilendirilir. Anahtarlar sözlükte benzersiz olmalıdır. Değerler çoğaltılabilir, ancak her değer benzersiz bir anahtarla ilişkilendirilir.

Sözlük Oluşturma

Sözlük nesneleri küme ayraçları { } kullanılarak oluşturulur.

Syntax:

dict = {key1 : value1, key2 : value2, . . . keyN : valueN }

Boş küme ayraçları kullanarak boş bir sözlük oluşturabilirsiniz.

dict = {}

Örnek

```
#create a dictionary
```

phone = { 'Evren':7445167, 'Ana':6413354,'Enes':6543210}

Sözlük Değerlerine Erişme

Alt simge operatörü([]), bir anahtarla ilişkili değeri döndürmek için kullanılır.

Sözlük, liste gibi dizi türünde bir kapsayıcı değildir, bu nedenle alt simge operatörü kullanılsa da, öğelere dizine/konuma göre erişemezsiniz.

Verilen anahtar sözlükte olmalıdır, değilse bir KeyError ortaya çıkacaktır. Erişmeden önce anahtar değerlerinin var olup olmadığını kontrol etmek için in/not in kullanın.

Syntax: dict[key] –> returns the value associated with a given key.

Sözlük Değerlerine Erişme

Örnek

phone = { 'Evren':7445167,

'Ana':6413354,

'Enes':6543210}

Sözlük Değerlerine Erişme

```
name = input('Enter name to search:')
while name != 'quit':
if name in phone:
print(name," 's contact number
is:",phone[name])
else:
print(name,' not in dictionary')
name = input('Enter name to search:')
```

Sözlük Değerleri İstisnalarına Erişme

Belirli bir anahtar sözlükte olmalıdır, değilse bir KeyError ortaya çıkacaktır. Erişmeden önce anahtar değerlerin mevcut olup olmadığını kontrol etmek için in/not in kullanabilirsiniz. Alternatif bir çözüm, Python özel durum işleme mekanizmasını kullanmaktır.

Syntax:

try:

```
#do this
```

except:

```
#code to execute if statement in try block throws exceptio
```

while name != 'quit':

try:

print(name,"'s contact number is:",phone[name])

Değer Ekleme/Güncelleme

Sözlükler değiştirilebilir , içeriğini değiştirdikten sonra değiştirebilirsiniz. Belirli bir anahtarla ilişkili bir değeri değiştirmek için, mevcut bir anahtardaki operatörü kullanarak yeni bir değer ayarlayın.

dict[key]= new_value

Sözlüğe öğe eklemek için, yeni bir benzersiz anahtar kullanarak yeni değeri belirtmeniz yeterlidir.

dict[new_key]= new_value

Değer Ekleme/Güncelleme

```
name = input('Enter person to update:')
number = input('Enter phone number:')
if name in phone:
phone[name] = number
print(name, 'updated! (', phone[name],')')
else:
phone[name] = number
print(name, 'added! (', phone[name],')')
```

Öğeleri Kaldırma Pop()

Sözlükten bir anahtar / değer çiftini kaldırmak için, pop ()

Syntax:

dict.pop(key) –> removes the key/value pair with the given key

Örnek:

```
#remove keys from dictionary
```

name = input('Enter person to remove:')

if name in phone:

phone.pop(name)

print(name, 'removed!')

else:

print(name, 'not in phone book')

Sözlükte Geçiş (Yineleme)

Sözlük, öğelerini verimlilik için optimize edilmiş bir sırada saklar ve bu, eklendikleri sıra olmayabilir. for döngüsü kullanarak sözlükteki tek tek tuşları yineleyebilirsiniz.

Örnek:

```
#traversing a dictionary
```

print('Contact List:')

for people in phone:

print(people,phone[people])

Note:

Yukarıdaki örneğin bir sözlükteki öğeler aracılığıyla yinelemeyi göstermek için kullanıldığını, ancak for döngüsü kullanılmadan da yapılabileceğini unutmayın.

Sözlükte Fonksiyonu Özeti

Function	Purpose
len(d)	d'deki öğelerin sayısını verir.
d.keys()	d'deki tuşların bir görünümünü döndürür.
d.values()	d'deki değerlerin bir görünümünü döndürür.
k in d	k d içindeyse true döndürür.
d[k]	d içindeki öğeyi k anahtarıyla döndürür.
d.get(k,v)	k d'deyse d[k], aksi takdirde v döndürür.
d[k]=v	v değerini d'deki k anahtarıyla ilişkilendirir, zaten k ile ilişkili bir değer varsa, bu değer değiştirilir.
d.pop(k)	Anahtar/değer çiftini verilen anahtarla k.
for k in d	d'deki tuşları yineler.

Sözlük Örneği

```
Sevil Degirmenci : 265332
        717488 Selahattin Bardakci ( +90 242 023 4313 )
        600194 Muge Tiryaki ( +90 242 577 439 )
Mansur Binici : 379795
        965441 Rasim Karga ( +90 242 089 9441 )
Bunyamin Aksoy : 213286
        486976 Cemre Degirmenci ( +90 242 107 3041 )
        695664 Yeter Demirci ( +90 242 341 2984 )
Ceren Avci : 238200
        556805 Fidan Badem ( +90 242 116 4943 )
        916757 Belma Koc ( +90 242 147 3348 )
        128269 Mahmut Terzi ( +90 242 388 8937 )
Erkan Marangoz : 506263
        612160 Mahmut Terzi ( +90 242 132 4672 )
Nejla Koc : 442171
        445691 Sami Tiryaki ( +90 242 159 6412 )
        200030 Berrak Ekmekci ( +90 242 432 5372 )
Ezgi Uzun : 297021
        502803 Asli Kartal ( +90 242 204 0806 )
        181651 Ozturk Nacar ( +90 242 426 4289 )
Hasip Burakgazi : 730083
        151659 Munire Macar ( +90 242 216 8125 )
Tuncay Sadik : 925031
        202773 Necla Peynirci ( +90 242 276 0402 )

Enter patient name to search: Mahmut Terzi
128269 Mahmut Terzi +90 242 388 8937 Doctor: Ceren Avci
612160 Mahmut Terzi +90 242 132 4672 Doctor: Erkan Marangoz
```

Figure 4: Sözlük Örneği

Sözlük Örneği

```
def read_doctors( file ):
doc_dict = {}

for line in file:

    line = line.split(',')

    line[-1] = line[-1][:-1]

    doc = tuple(line[0:2])

    patient = tuple(line[2:])
```

Sözlük Örneği

```
if doc in doc_dict:
        doc_dict[doc].append(patient)
    else:
        doc_dict[doc] = [patient]
return doc_dict
```

Sözlük Örneği

```
def display_data( pname, doc_dict ):
for key in doc_dict:
for patient in doc_dict[key]:
if patient[1].lower() == pname.lower():
print(patient[0],pname,patient[2],'Doctor:',key[1])
```

Sözlük Örneği

```
file = open('patient_list.txt', 'r')
dict1 = read_doctors(file)
file.close()
```

Sözlük Örneği

for doc in dict1:

```
print(doc[1],':',doc[0])

for patient in dict1[doc]:

    print('\t',patient[0],patient[1],'(',patient[2],')')
#Input the name of a patient and display their doctor.
```

name = input(‘Enter patient name to search:’) display_data(name, dict1)

Bellekteki Listeler ve Sözlükler

Listeler ve sözlükler değiştirilebilir Değiştirilebilir türler, değişmez türlerden farklı davranır Yapılandırılmış nesne bellekte saklanır ve değişken adı nesneye işaret eder (başvurur) Bir nesne değiştirildiğinde, o nesneye işaret eden (nesneye başvuran) herhangi bir değişken de etkilenir Buna 'yan etkiler' denir, yani bir değişkenin değiştirilmesi diğer değişkenleri etkileyebilir.

Alias (Takma ad)

Aynı nesneye başvuran iki veya daha fazla başvuruya birbirinin takma adı denir Bu ilginç bir durum yaratır Birden çok referans değişkeni kullanılarak bir nesneye erişilebilir Takma adlar yararlı olabilir, ancak dikkatli bir şekilde yönetilmelidir Bir nesneyi tek bir başvuru aracılığıyla değiştirmek, onu tüm takma adları için değiştirir, çünkü gerçekte yalnızca bir nesne vardır.

Alias (Takma ad) Örneği

```
1  a = 1
2  b = a
3  print(a)
4  print(b)
5
6  warm = ['red', 'yellow', 'orange']
7  hot = warm
8  hot.append('pink')
9  print(hot)
10 print(warm)
```

```
1
1
['red', 'yellow', 'orange', 'pink']
['red', 'yellow', 'orange', 'pink']
```

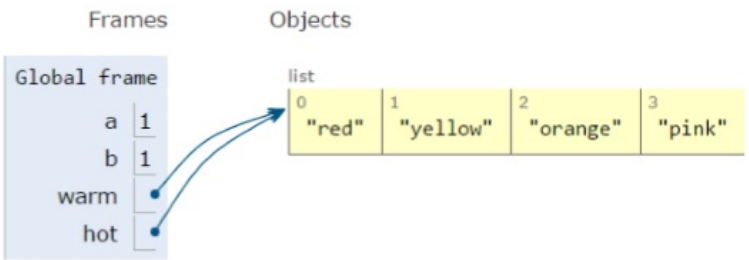

Figure 5: Alias (Takma ad)

Yukarıdaki örnekte sıcak, ısının birinin diğerini değiştirmesi için bir takma addır! Bu nedenle, append() fonksiyonunun bir yan etkisi vardır, bunu sıcak darbelere uygular ılık a ve b skaler değerler olduğundan, takma adlar/yan etkiler yoktur.

Klonlama

Önceki örnekte gördüğünüz gibi, başka bir değişkene bir nesne atadığınızda, bir kopya değil, bir diğer ad oluşturur. Bir nesneyi klonlamak, mevcut bir nesneden, bu durumda bir listeden verileri kopyalayarak yeni bir nesne oluşturmayı içerir. Bir listeyi klonlamanın iki yolu vardır: yerleşik bir işlev olan list() kullanarak veya dilimlemeyi kullanarak.

Klonlama Örneği

```
1 cool = ['blue', 'green', 'grey']
2 chill = cool[:]
3 chill.append('black')
4 print(chill)
5 print(cool)
```

```
['blue', 'green', 'grey', 'black']
['blue', 'green', 'grey']
```

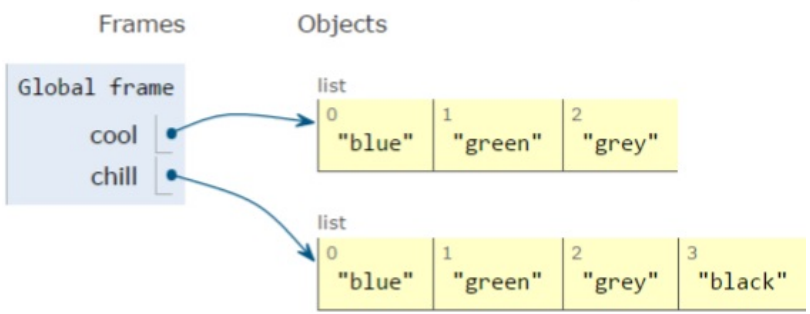

Figure 6: Alias (Takma ad)

list komutunu kullanarak yeni bir liste oluşturun chill list(cool)

Mutasyon ve İterasyon

Üzerinde yineleme yaparken bir listeyi değiştirmekten kaçının

def remove_dups (L1,L2):

for e in L1:

```
if e in L2:

  L1.remove(e)
```

L1=[1,2,3,4]

L2=[1,2,5,6]

Mutasyon ve İterasyon

```
def remobe_dups(L1,L2):
L1_copy=L1[:]
for e in L1_copy:
if e in L2:

  L1.remove(e)
```

Mutasyon ve İterasyon

L1 [2,3,4] değil [3,4] Neden? Pyhon, dizini takip etmek için dahili bir sayaç kullanır, döngüdedir, mutasyon liste uzunluğunu değiştirir, ancak Python sayaç döngüsünü güncellemez, element2'yi asla görmez.

İşlev parametreleri aşağıdakileri izlerken listeler

```
from random import randrange
def generate_list(n):
my_list = []
for i in range(1,n+1):
my_list.append(randrange(1,101))
return my_list
```

İşlev parametreleri aşağıdakileri izlerken listeler

```
def double_list(my_list):
for i in range(len(my_list)):
my_list[i] = my_list[i] * 2
def double_value(x):
x = x / 2;
print(x)
```

İşlev parametreleri aşağıdakileri izlerken listeler

```
list_one = generate_list(5)

print(list_one)

double_list(list_one)

print(list_one)

double_value(list_one[0])

print(list_one)
```

Dizi Türlerinde Sık Yapılan İşlemler (str, tuple, list)

seq[i] Dizideki i'inci öğeyi döndürür

len(seq) Dizinin uzunluğunu verir

seq1 + seq2 İki dizinin birleşimini verir

n * seq Ardışık N kez tekrarlanan bir dizi döndürür.

seq [star:end] Dizinin bir dilimini döndürür

e in seq Dizide e yer alıyorsa True, aksi takdirde False olur

e not in seq e dizide değilse True, aksi takdirde False

for e in seq Dizinin öğeleri üzerinde yinelenir.

Dizi Tiplerinin Karşılaştırılması

Type	Type of elements	Examples of literals	Mutable
`str`	characters	`''`, `'a'`, `'abc'`	No
`tuple`	any type	`()`, `(3,)`, `('abc', 4)`	No
`list`	any type	`[]`, `[3]`, `['abc', 4]`	Yes

Figure 7: Dizi Tiplerinin Karşılaştırılması

Tablolar – Liste Listeleri

Bir tablo veya matris, değerlerin satır ve sütunlarından oluşan bir düzenlemedir. Bazen programlarımızda veri tablolarını/matrislerini saklamak gerekir (örneğin bilimsel veya finansal uygulamalar). Python'un tablo oluşturmak için bir veri türü yoktur, ancak Python listeleri kullanılarak iki boyutlu bir tablo yapısı oluşturulabilir.

Tablo Oluşturma

table = [[0,3,0],

[0,0,1],

[2,0,3],

[3,1,0],

[2,5,4]]

Tablo Oluşturma

```
table = []
table.append([0,3,0])
table.append([0,0,1])
table.append([2,0,3])
table.append([3,1,0])
table.append([2,5,4])
```

Tablonun Bir Öğesine Erişme

Tablodaki belirli bir öğeye erişmek için, sırasıyla satır ve sütunu seçmek üzere ayrı köşeli parantez içinde iki dizin değeri belirtmeniz gerekir. Tablodaki satır sayısını hesaplamak için len(table) Tablodaki belirli bir satırın uzunluğunu hesaplamak için len(table[row])

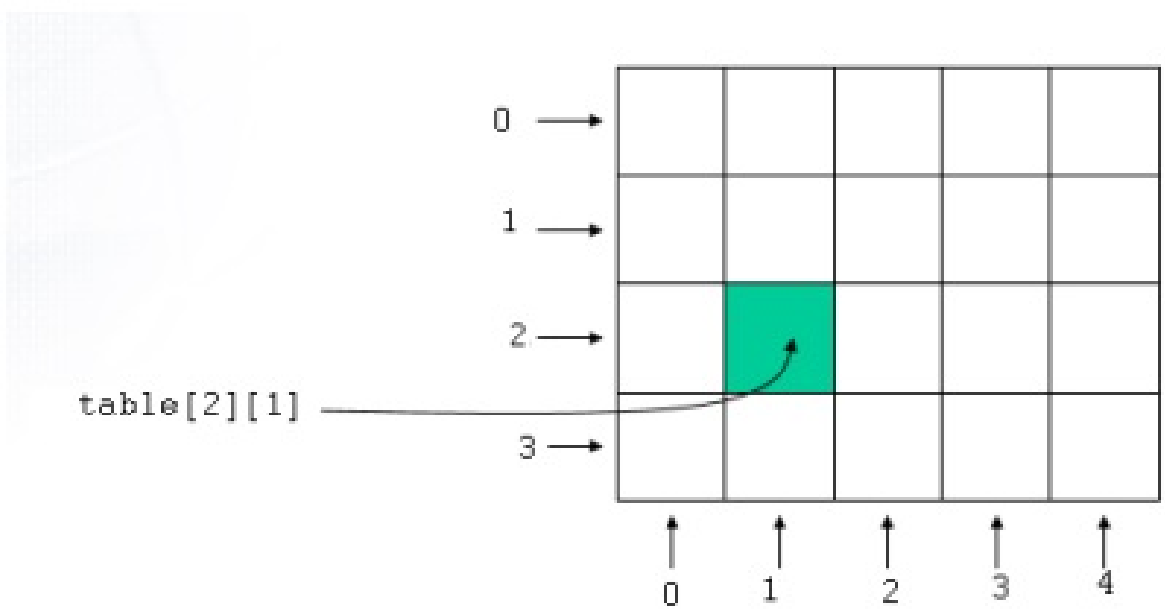

Figure 8: Tablo Oluşturma

Tablonun Bir Öğesine Erişme

Bir tablodaki tüm öğelere erişmek için iç içe geçmiş 2 döngü kullanırsınız. Öğelere erişmek için range işlevini veya in operatörünü kullanabilirsiniz

for row in table:

for col in row:

print(col,end=" ")

print()

Tablonun Bir Öğesine Erişme

```
for row in range(len(table)):
for col in range(len(table[row])):
print(table[row][col],end=" ")
print()
```

Alıştırmalar

1. Aşağıdakileri yapan bir program yazın
   - Kullanıcıdan bir değer tablosu girin (2x3)
   - print_table() yöntemi kullanıldığında tablo görüntülenir.
   - sum_rows() yöntemini kullanarak, her satırın toplamını görüntüler.
   - sum_cols() yöntemini kullanarak, her sütunun toplamını görüntüler.

Alıştırmalar

2. Parametre olarak bir sözcük tablosu ve bir dize sözcüğü alan ve sözcüğün geçtiği tüm yerleri bir yıldızla değiştiren bir işlev yazın.
3. Kullanıcı -1 girene kadar değerleri girin ve bir listede saklayın. Ardından bir limit girin ve listede limiti aşan ilk öğenin dizinini görüntüleyin ve o elemanı kaldırın.
4. Kullanıcıdan 5 kelime girin ve bir listede saklayın.
   - Listenin sonundan başlayarak, büyük harfle başlayan tüm Dizeleri görüntüleyin.
   - Listedeki en kısa kelimeyi görüntüleyin.
5. İki tuples kesişimini bulmak için bir işlev yazın

Alıştırmalar

6. n1 ve n2'nin pozitif tam sayılar olduğunu varsayarak, 1'den büyük en küçük ortak böleni ve n1 ve n2'nin en büyük ortak bölenini bulmak için bir fonksiyon yazın. Ortak bölen yoksa, (Yok, Yok) değerine sahip iki öğeli bir tanımlama grubu döndürün.