Paralel Hesaplama

Öğr. Gör. Zafer SERİN

LİSTELER

- Listeler sıralı elemanlar dizisidir. Python'da [] ile veya list() constructorı ile liste tanımlanabilir.

```
liste = ["deneme", False, 2.5, 5]
```

```
liste1 = list(["deneme", False, 2.5, 5])
```

- Liste elemanlarına stringlerde olduğu gibi indisleri ile erişilebilir. İndisler 0'dan başlar ve son elemana kadar(listenin uzunluğunun 1 eksiği) gider. Elemanlara tersten erişmek içinde -1 indisi ile başlanarak bu sefer sola doğru -2, -3 … şeklinde ulaşılabilir.

LİSTELER

- Örnek:

```
liste = ["deneme", False, 2.5, 5]
print(liste[1])
print(liste[-1])
```

```
False
5
```

LİSTELER

- Listelerin elemanları değiştirilebilir(mutable). Stringlerin elemanları değiştirilemez!(immutable)

```
liste = ["deneme", False, 2.5, 5]
liste[0] = "test"
```

Çalışır

```
isim = "Zafer"
isim[1] = "e"
```

Çalışmaz!

ÖNEMLİ LİSTE METOTLARI

- append(): listenin sonuna eleman ekler
- clear(): listeyi tamamen boş hale getirir
- copy(): listeyi kopyalar
- count(): listedeki bir elemanın sayısını verir
- extend(): listeyi başka bir eleman ile genişletir
- index(): listedeki bir elemanın indisini verir
- insert(): listenin belirli bir indisine eleman eklemeyi sağlar
- pop(): varsayılan olarak liste sonundan eğer verilirse indisten değeri siler
- remove(): verilen değeri listeden siler

ÖNEMLİ LİSTE METOTLARI

- reverse(): listeyi ters çevirir
- sort(): listeyi sıralar. Varsayılan olarak sayısal listeyi küçükten büyüğe metinsel listeyi harf sırasına göre sıralar. Hem metinsel hem sayısal ifadelerin olduğu listeleri sıralayamaz! Not: boolean elemanlarıda sıralayabilir False için 0 sayısal değerini True için 1 sayısal değerini kullanır.

LİSTELER İLE STACK VE QUEUE

- Listelerin sonuna eleman ekleyip(append()) sonundan eleman çıkarılırsa(pop()) LIFO(Last in First Out – Son gelen ilk çıkar) yöntemi kullanılarak stack(yığın) veri yapısı oluşturulabilir.
- Listelerin sonuna eleman ekleyip(append()) başından eleman çıkarılırsa(pop(0)) FIFO(First in First Out – İlk gelen ilk çıkar) yöntemi kullanılarak queue(kuyruk) veri yapısı oluşturulabilir.

LİSTELER İÇİNDE LİSTE OLUŞTURMA

- Liste içinde liste tanımlanabilir ve indisler ile ulaşılabilir.

```
liste = [1, "selam", [1, 2, "deneme", True], 2.4, False]
print(liste[2][1])
```

2

DEMETLER(TUPLES)

- Tuple, sıralı değişmez nesneler dizisidir. Python'da () ile veya tuple() constructorı ile tuple tanımlanabilir. Tuple elemanları doğrudan değiştirilemez! Listelere göre daha hızlıdırlar.

```
demet = (1, 2, "selam", False, 3.4) # tuple tanımlama
demet1 = tuple((1, 2, "deneme", True)) # tuple tanımlama
print(demet1[3]) # elemanın indisi ile okunabilir
demet(0) = 5 # !değişiklik yapılamaz hata verir
demet[0] = "ornek" #!değişiklik yapılamaz hata verir
```

ÖNEMLİ TUPLE METOTLARI

- Tuple elemanları doğrudan değiştirilemediği için 2 adet önemli metota sahiptir.
- count(): Bir elemanın tupleda kaç tane olduğunu verir.
- index(): Bir elemanın indisini verir.

KÜMELER(SETS)

- Set, sırasız tekil elemanlar topluluğudur. Python'da {} ile veya set() contructorı ile set tanımlanabilir. Matematiksel kümeler olarak düşünebilir. Set içerisinde aynı elemandan birden çok olamaz! Set elemanları sıralı olmak zorunda değildir dolayısıyla set elemanlarına erişim için indis kullanılamaz!

KÜMELER(SETS)

- Örnek:

```
kume = {1, "test", "ornek", False, 5.4} # set tanimi
kume1 = set({1, 3, 6, 2.3, "selam", True}) # set tanimi
print(kume) # set elemanlari sirali değildir!
```

{False, 1, 'ornek', 5.4, 'test'}

```
print(kume[0]) #set elemanlarina indis ile erişilemez!
```

ÖNEMLİ SET METOTLARI

- add(): Set'e eleman ekler.
- clear(): Set'i tamamen temizler
- copy(): Set'i kopyalar
- difference(): İlgili Set ile bir başka Set'in farkını alır
- difference_update(): İlgili Set ile bir başka Set'in farkını alır ve günceller
- discard(): Set'teki verilen elemanı siler
- intersection(): İlgili Set ile bir başka Set'in kesişimini alır
- intersection_update(): İlgili Set ile bir başka Set'in kesişimini alır ve günceller

ÖNEMLİ SET METOTLARI

- isdisjoint(): 2 setin kesişiminin boş olup olmadığını sorgular
- issubset(): Bir Set'in bir başka Set'in altkümesi olup olmadığını sorgular
- İssuperset(): Bir Set'in bir başka Set'i kapsayıp kapsamadığını sorgular
- pop(): Set yapısı sıralı olmadığından rastgele bir elemanı siler
- remove(): Verilen elemanı Set içinden siler
- symmetric_difference(): 2 Set arasındaki simetrik fark işlemini gerçekleştirir.
- symmetric_difference_update(): 2 Set arasındaki simetrik fark işlemini gerçekleştirir ve günceller

ÖNEMLİ SET METOTLARI

- union(): 2 Set için birleşim işlemini gerçekleştirir.
- update(): Set için güncelleme işlemini yapar.

SÖZLÜKLER(DICTIONARIES)

- Dictionary, birbirlerinden virgüllerle ayrılmış ‘anahtar:değer’/’key:value’ şeklinde eşleşen veri yapılarıdır. Python’da {} veya dict() constructorı ile dictionary tanımlanabilir. Sözlük değerleri değiştirilebilir.

```
sozluk = {"isim":"Zafer", "yas": 29}
print(sozluk["yas"])
```

29

ÖNEMLİ DICTIONARY METOTLARI

- clear: Dictionary'i tamamen temizler
- copy: Dictionary'i kopyalar
- fromkeys: Belirli bir anahtar kümesinden yeni bir dictionary oluşturur
- get: İlgili key(anahtar) değerinin karşılığı değeri(value) getirir
- items: Dictionary'de bulunan tüm anahtar-değer ikililerini verir
- keys: Dictionary'de bulunan anahtar(key) değerleri verir

ÖNEMLİ DICTIONARY METOTLARI

- pop: Dictionary'den verilen anahtarı sahip değeri siler
- popitem: Dictionary'nin en sonundaki anahtar-değeri çıkarır
- setdefault: Dictionary'de varsa anahtarın değerini yoksa default değeri getirir.
- update: Dictionary'i başka bir dictionary ile güncellemek için kullanılır
- values: Dictionary'de bulunan value(değer) ları gösterir.