Python 201

1.Tuple

Bunlar da listeler gibi veri yapılarıdır.Örneğin,

tuple_adi = (“ali” , ”veli”,1,2,3.2,[1,2,3,4,5]) veya

tuple_adi = “ali” , ”veli”,1,2,3.2,[1,2,3,4,5] şeklinde oluşturulabilir.

Tuple da aslında liste.Listelerden tek farkı değiştirilemez olması.

elemanlara erişme işlemleri listelerle aynıdır. tuple_adi[0] gibi 0.indekse(1.elemana) erişilebilir.

Listeler ve Tuple’ın farkı ;

Listeler

- Kapsayıcıdır (içinde her değişkeni atabilirsin)
- Değiştirilebilir
- Sıralıdır(1.eleman,2.eleman)

Tuple

- Kapsayıcıdır
- Değiştirilemez
- Sıralıdır

2.Sözlükler

Anahtar değer(key)ler ile değerlerinin(value) bir arada tutulduğu veri yapısıdır.Yani her keye karşılık bir value değeri olacak.Listelerde olduğu gibi indeks işlemi yapılamaz.Yani sozluk_adi[0] gibi elemana erişemiyoruz.key değerine göre elemana erişcez.

Kapsayıcı

Sırasız

Değiştirilebilir

Örnek:

```
sozluk = {
    ”REG” : “Regresyon Modeli”,
    “LOJ” : “Lojistik Regresyon”,
    “CART” : “Classification and Reg”
}
```

burada key değerleri REG,LOJ,CART bunların karşılarındaki değerler ise value değerleri

Elemanlara erişmek için sozluk[0] diye kullanamıyoruz.Örneğin REG keyindeki değere erişmek için sozluk[”REG”] şeklinde kullanabiliriz.

Sözlük içinde sözlük yapısı kurulabilir.

```
sozluk2 = {
    ”REG” : {
        “RMSE” : 10,
        “MSE” : 20,
        “SSE” : 30
      },
    “LOJ” : “Lojistik Regresyon”,
    “CART” : “Classification and Reg”
}
```

bunun içerisindeki SSE’e erişmek için

sozluk2[”REG”][”SSE”] şeklinde kullanmalıyız.

Sözlüklerde Eleman Ekleme Ve Değiştirme

sozluk = {

”REG” : “Regresyon Modeli”,

“LOJ” : “Lojistik Regresyon”,

“CART” : “Classification and Reg”

}

sözlüğüne GBM keyinde eleman eklemek

sozluk[”GBM”] = “Gradient Boosting Mac” şeklinde.

Bu şekilde en sona GBM elemanını ekleyecek

Değiştirmek için ise key değerine ulaşıp değiştircez.Örneğin,

sozluk[”REG”] = “Çoklu Doğrusal Regresyon”

Not: Sözlükte hata alıyorsan büyük ihtimalle key error hatasıdır.

3.Setler

Sırasızdır

Değerleri eşsizdir

Değiştirilebilirdir

Amacı;

Programlamada hız istediğimiz ve elemanları eşşiz olacak bir şeyler istediğimizde kullanılır.Matematikteki kümelere benzer.

Olay aslında bir liste verdiğimizde o listede eğer aynı elemanlar varsa onların sadece 1 tanseini alıyor.Yani her bir elemandan sadece 1 tane oluyor.Bu da performans(hız) kazandırıyor.

liste = [”ali” , ”lütfen” , ”topa” , ”bakma” , ”uzaya” , ”git” , ”git”, ”ali” , ”git”]

s = set(liste)

s → {”ali” , “ata” , “bakma” , “git” , “lütfen” , “uzaya”}

Her bi elemanı en çok 1 tane aldı ve alfabetik olarak sıralı bir şekilde set oluşturuldu.

Setler sırasız olduğu için elemanlarına set[1] gibi erişemiyoruz.

setlerde eleman ekleme işlemi set_adi.add(“eklenecek şey”) şeklindedir.

setlerde eleman silme işlemi set_adi.remove(“silinecek şey”) şeklindedir.

eleman silerken eğer eleman o sette yoksa hata verecek.Hem sil hem de eğer o eleman yoksa hata verme devam et istersek set_adi.discard(“silinecek şey”) fonksiyonunu kullanabiliriz.

Setlerde Küme İşlemleri

- difference() veya - işareti : iki kümenin farkını verir

  set1 = set([1,3,5])

  set2 = set([1,2,3])

  set1.difference(set2) → set1 fark(difference) set2 demek

  set2.difference(set1) → set 2 fark set1 demek

- symmetric_difference() : iki kümede de olmayanları verecek

- intersection() veya & işreti : iki kümenin kesişimi ifadesi

  set1.intersection(set2) → set1 ve set2 kümelerinin kesişim kümesi

- union() iki kümenin birleşimi

  set1.union(set2) → set1 ve set2 kümelerinin birleşimi

Setlerde Sorgu İşlemleri

Kesişimin boş olup olmadığı,bir küme diğerini kapsar mı gibi sorgu işlemlerini yapmaktır

Örnekler

1-İki kümenin kesişiminin boş olup olmadığına bakan fonksiyon

set1.isdisjoint(set2) → true veya false döner

2-Bir kümenin diğer kümenin alt kümesi mi kontrolü;

set1.issubset(set2) → set1 , set2 nin alt kümesi mi kontrolü

3-Bir küme diğer kümeyi kapsıyor mu kontrolü;

set2.issuperset(set1) → set2 , set1'i kapsıyor mu?