

Python for Data Science

Çalışma Dökümanı

Recep Aydoğdu

İçindekiler

[Data Science Kullanılan Alanlar 5](#_Toc37001710)

[Data Science Proje Döngüsü 5](#_Toc37001711)

[Python Programlama 6](#_Toc37001712)

[Temel Hareketler 6](#_Toc37001713)

[Integer, Float ve String 6](#_Toc37001714)

[String Metodları 6](#_Toc37001715)

[Veri Yapıları (Data Types) 8](#_Toc37001716)

[Listeler 8](#_Toc37001717)

[Liste Elemanlarına Ulaşma 9](#_Toc37001718)

[Liste İçi Type Sorgulama 9](#_Toc37001719)

[Liste elemanlarını değiştirme 9](#_Toc37001720)

[Listeye eleman ekleme 9](#_Toc37001721)

[Listeden eleman silme 9](#_Toc37001722)

[append ve remove metodlari 10](#_Toc37001723)

[insert metodu = 10](#_Toc37001724)

[pop metodu 10](#_Toc37001725)

[count metodu 10](#_Toc37001726)

[copy metodu 10](#_Toc37001727)

[extend metodu 11](#_Toc37001728)

[index metodu 11](#_Toc37001729)

[reverse metodu 11](#_Toc37001730)

[sort metodu 11](#_Toc37001731)

[clear metodu 11](#_Toc37001732)

[Tuple (Demet) 12](#_Toc37001733)

[Tuple Oluşturma 12](#_Toc37001734)

[Eleman İşlemleri 12](#_Toc37001735)

[Dictionary (Sözlük) 12](#_Toc37001736)

[Dictionary Nedir? 12](#_Toc37001737)

[Dictionary Oluşturma 12](#_Toc37001738)

[Eleman Seçme İşlemleri 13](#_Toc37001739)

[Eleman Ekleme & Değiştirme 13](#_Toc37001740)

[Sets (Kümeler) 14](#_Toc37001741)

[Set Oluşturma 14](#_Toc37001742)

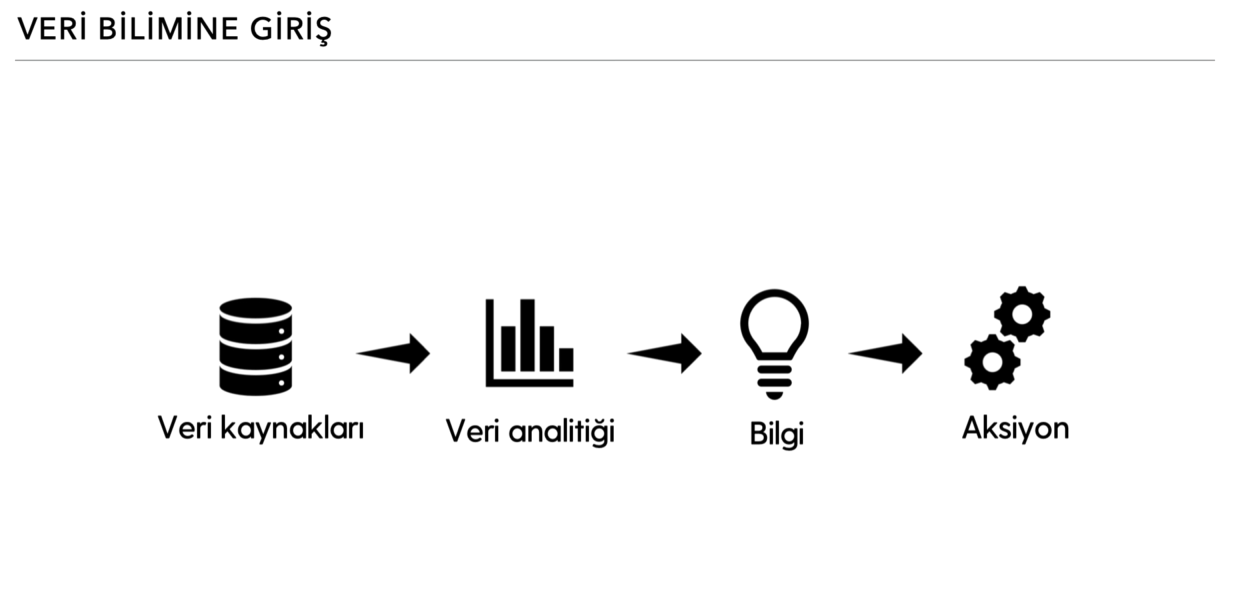
[Set’lere eleman ekleme ve çıkarma işlemleri 15](#_Toc37001743)

[Set’lerde Fark İşlemleri 17](#_Toc37001744)

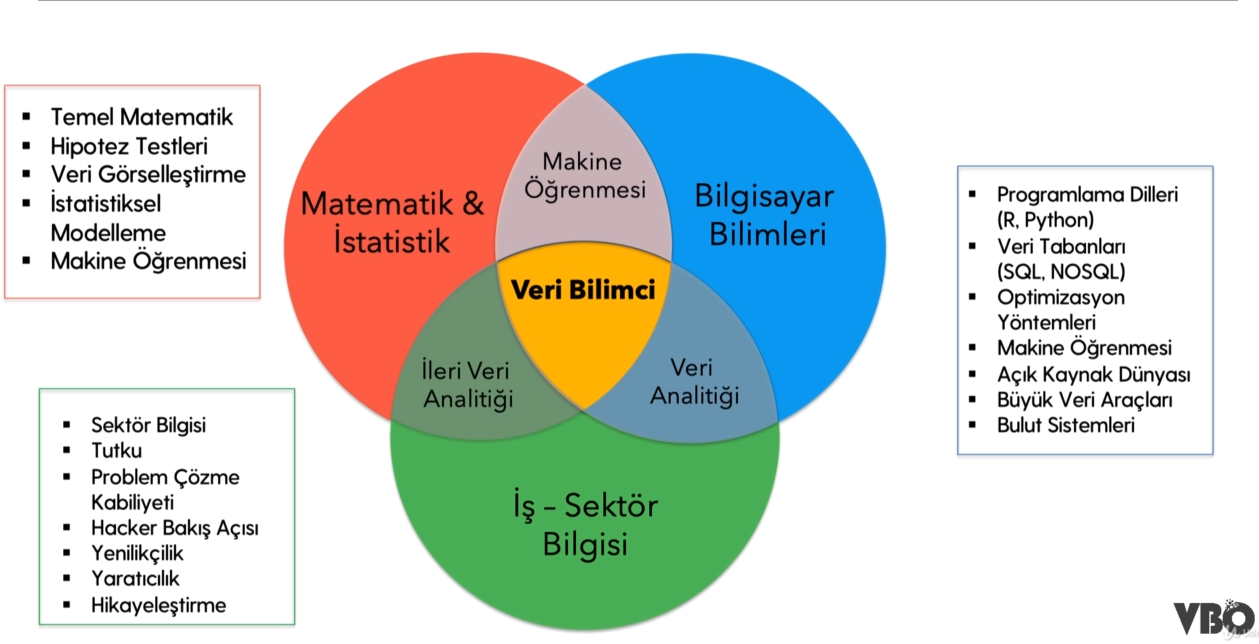
[Set’lerde Kesişim ve Birleşim İşlemleri 17](#_Toc37001745)

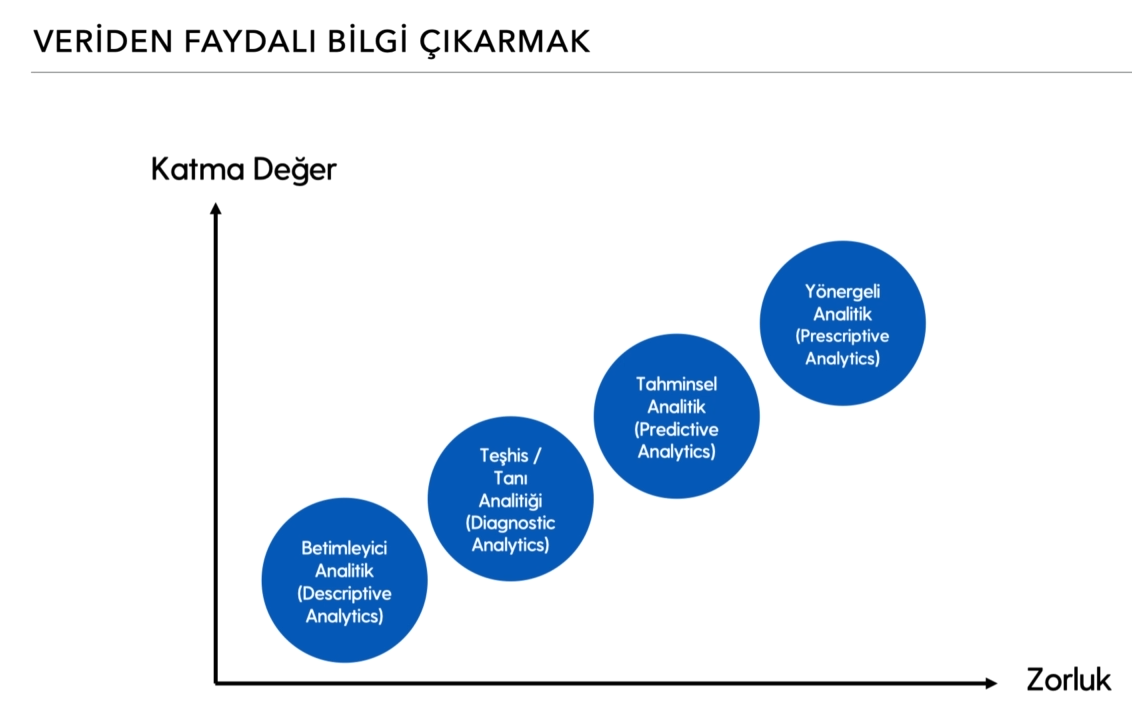
[Set’lerde Sorgu İşlemleri 18](#_Toc37001746)

**Data Science**

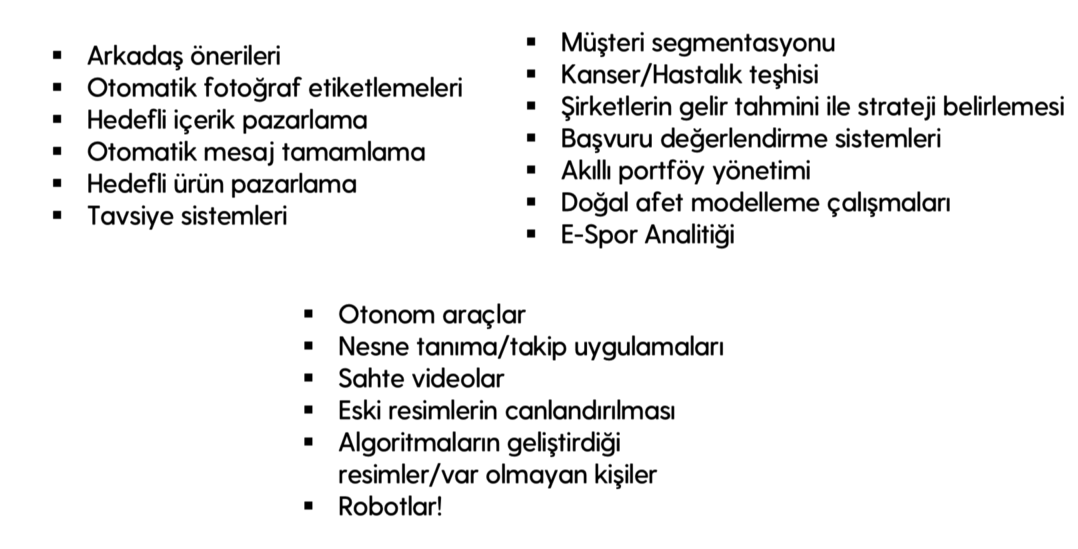


Veri Bilimci, veriden faydalı bilgi çıkarma sürecini yöneten kişidir.

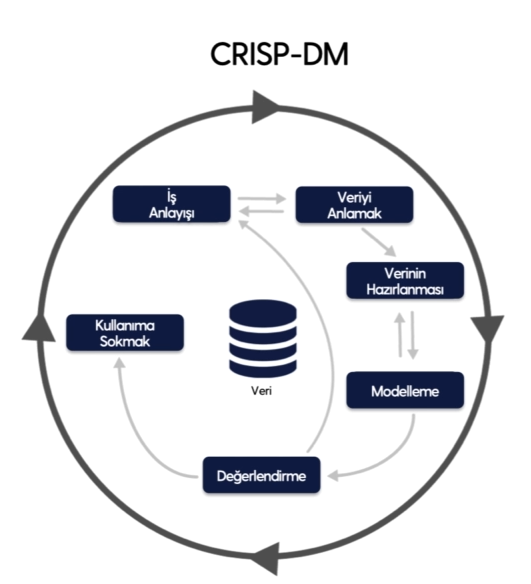




# Data Science Kullanılan Alanlar



# Data Science Proje Döngüsü



# Python Programlama

## Temel Hareketler

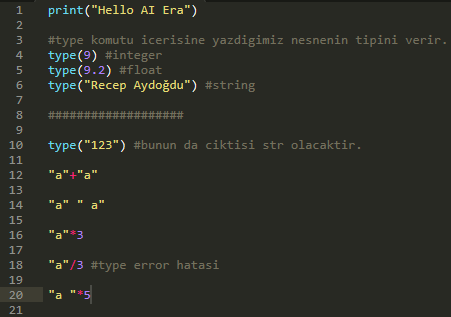
### Integer, Float ve String

**Integer =** 9 gibi ondalıksız sayılar.

**Float =** 9.2 gibi ondalıklı sayılar.

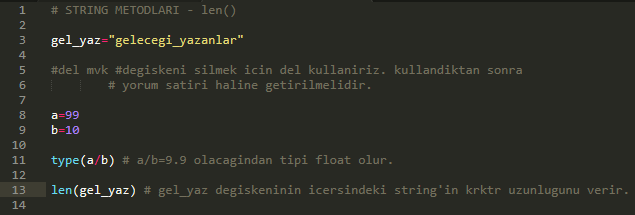
**String =** Karakter dizileri. “Çift tırnak” veya ‘Tek tırnak’ içinde yazılır.

**Type =** type() içersine yazılan nesnenin tipini verir.

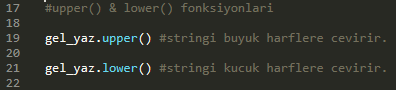


### String Metodları

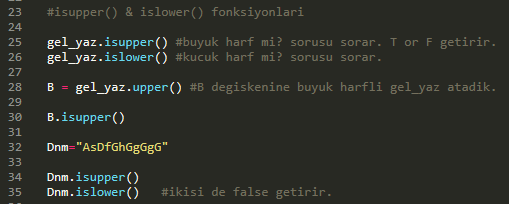
**len()=** içerisine yazılan değişkenin uzunluğunu verir.



**upper() & lower() =**



**isupper() & islower() =**



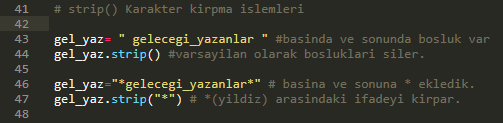
**replace() =**



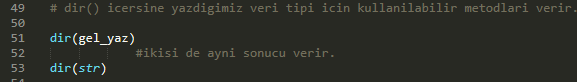
replace(“eski\_karakter”,”yeni\_karakter”)

gelecegi\_yazanlar 🡪gelecegi\_yızınlır

**strip() =** Karakter kırpma işlemleri



**dir() =**



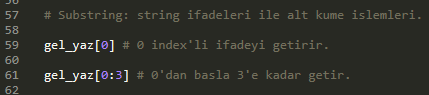
**capitalize() =** İlk harfi büyütür.

gel\_yaz.capitalized()

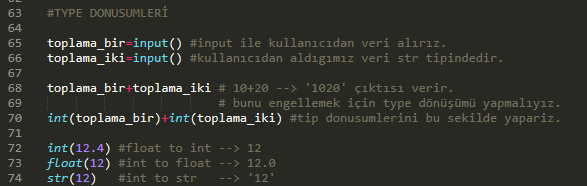
**title() =** Her kelimenin ilk harfini büyütür.

gel\_yaz.title()

**Substring =** Alt küme işlemleri



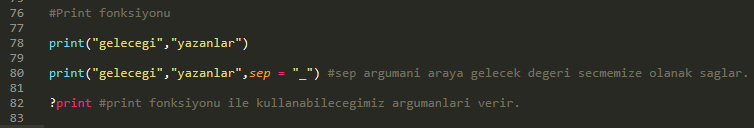
**Type Dönüşümleri**



**print() fonksiyonu**

print(“gelecegi”,”yazanlar”) 🡪 gelecegi yazanlar

print(“gelecegi”,”yazanlar”,sep = (“\_”)) 🡪 gelecegi\_yazanlar



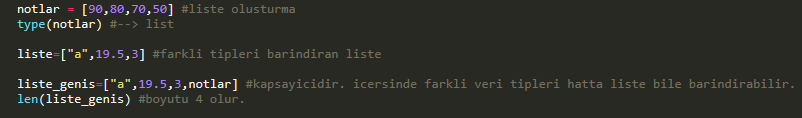
## Veri Yapıları (Data Types)

### Listeler

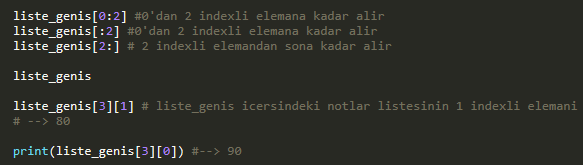
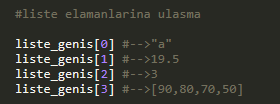
1. Değiştirilebilir
2. Kapsayıcıdır (Farklı tipte verileri tutabilir.)
3. Sıralıdır

Köşeli parantez [ ] ya da list() fonksiyonu ile liste oluşturabiliriz.

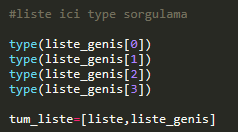
Liste bir üst type’dır içersinde farklı type’da veriler barındırabilir.



#### Liste Elemanlarına Ulaşma

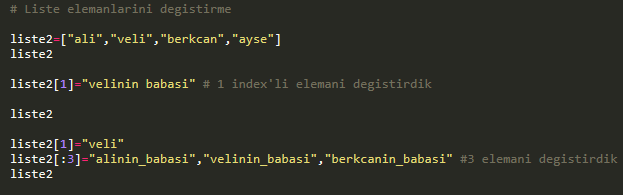


#### Liste İçi Type Sorgulama

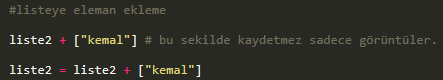


**del liste 🡪 liste’yi siler**

#### Liste elemanlarını değiştirme



#### Listeye eleman ekleme



#### Listeden eleman silme

del liste2[5] 🡪 5 index’li elamanı siler.

#### append ve remove metodlari

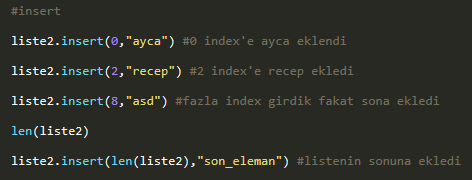
liste2.append("berkcan") 🡪sona ekleme yapar

liste2.remove("alinin\_babasi") 🡪silme yapar

liste2.remove("velinin\_babasi")

#### insert metodu

index’e göre ekleme yapar.



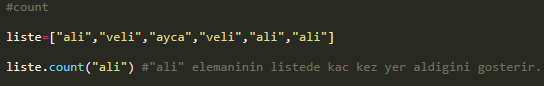
#### pop metodu

index’e göre silme yapar.

liste2.pop(0) #0 index degerli elemani siler

liste2.pop(1) #1 indexli elemani siler.

#### count metodu

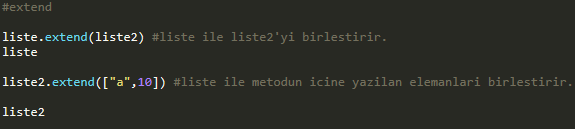


🡪 3

#### copy metodu

liste\_yedek=liste.copy() 🡪 liste'yi liste\_yedek'e kopyalar.

#### extend metodu

İki farklı listeyi birleştirir.

#### index metodu



#### reverse metodu

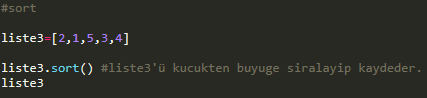
liste = [1,2,3]

liste.reverse() 🡪 liste elemanlarını ters sırayla kaydeder.

liste = [3,2,1]

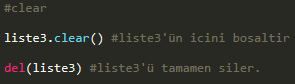
#### sort metodu

Elemanları küçükten büyüğe sıralar.



#### clear metodu

liste’nin içini boşaltır.



### Tuple (Demet)

1. Kapsayıcıdır
2. Sıralıdır
3. Değiştirilemez (Listeden farkı budur.)

#### Tuple Oluşturma



**NOT=** Tek elemanlı tuple oluştururken sonuna virgül koymalıyız. Aksi takdirde tuple oluşturmak istediğimiz anlaşılamaz.

Örneğin; t = (“eleman”,)

#### Eleman İşlemleri

Tuple’larda eleman işlemleri listeler ile birebir aynıdır. (index’e göre erişim vs.)

t=(1,2,3,4)

t[0] 🡪 1

t[-1] 🡪4 (sondan birinci eleman demektir.)

### Dictionary (Sözlük)

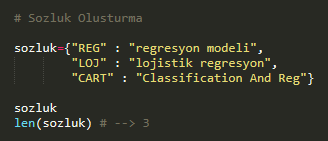
1. Kapsayıcıdır
2. Sırasızdır 🡪 Listelerden farkı budur.
3. Değiştirilebilirdir.

#### Dictionary Nedir?

Key’ler ve bu key’lerin karşılıklarının bir arada tutulduğu veri yapısıdır.

Listelerde olduğu gibi index’leme yapılmaz.

#### Dictionary Oluşturma



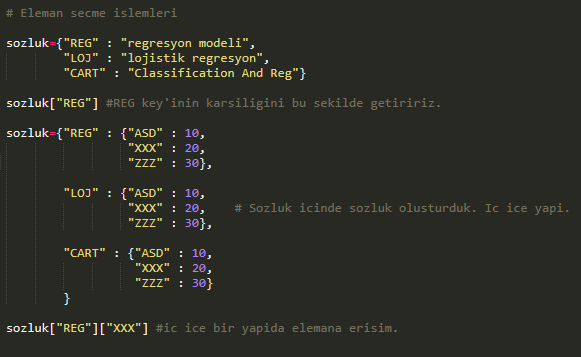
{“key” : “key’in karşılığı”}

**NOT=** Sözlüklerde key’ler sadece sabit veri yapılarından oluşabilir. list gibi yapılardan olamaz. String ve sayılar sabit ver yapılarıdır.

Sabit veri yapısı değiştirilemez demektir. Tuple’da buna dahildir.

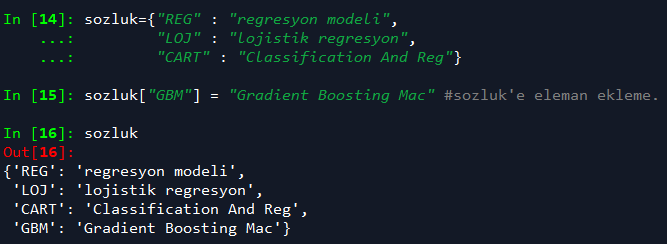
t = (“tuple”,) 🡪 sozluk = { t : “tuple’dan key olur” }

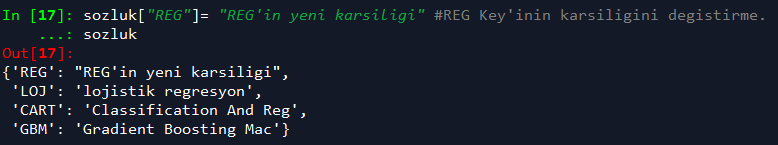
#### Eleman Seçme İşlemleri



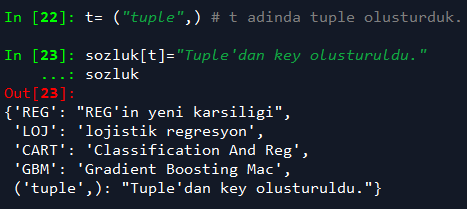


#### Eleman Ekleme & Değiştirme





REG key’i olmasaydı yeni key oluşturulacaktı.



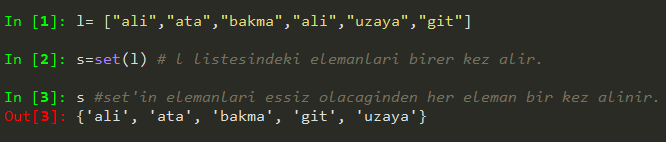
### Sets (Kümeler)

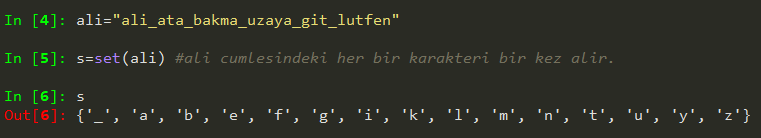
1. Sırasızdır (Index değerleri yok.)
2. Değerleri eşsizdir. (Tekrar eden değeri olmaz.)
3. Değiştirilebilir.
4. Kapsayıcıdır. Farklı türden veri yapıları barındırabilir.

Set’ler performans odaklı veri tipleridir. Programlama anlamında biraz daha hız istediğimizde kullanılır. Matematiksel anlamda bu veri yapıları kümelere benzer.

#### Set Oluşturma

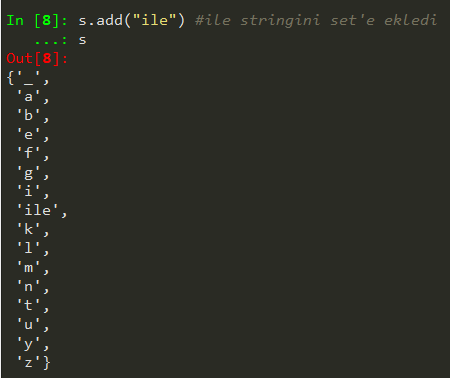
s = set() 🡪 s isminde bir set oluşturuldu.

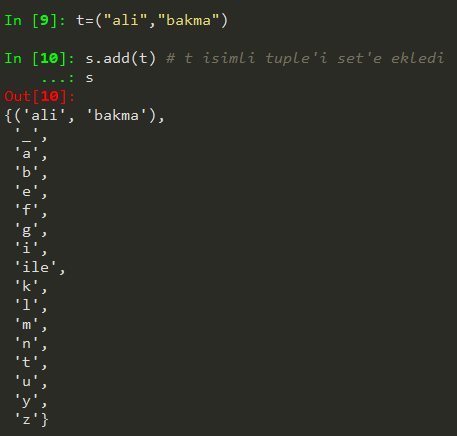


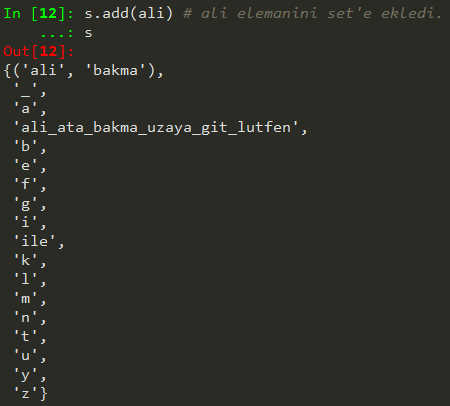


#### Set’lere eleman ekleme ve çıkarma işlemleri

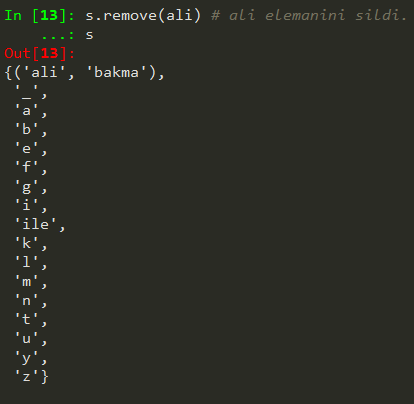
add() fonksiyonu ile ekleme yaparız.

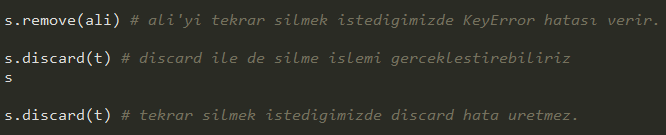






remove() fonksiyonu ile set’lerden eleman silebiliriz.

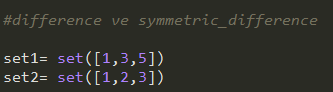




#### Set’lerde Fark İşlemleri

**difference & symmetric\_difference**

**difference =** kümelerin farkını verir.



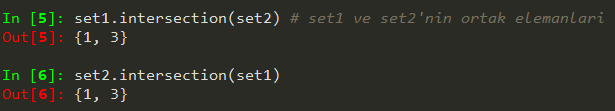
**symmetric\_difference =** ikisinde de ortak olmayan elemanları verir.



#### Set’lerde Kesişim ve Birleşim İşlemleri

**intersection & union & intersection\_update**

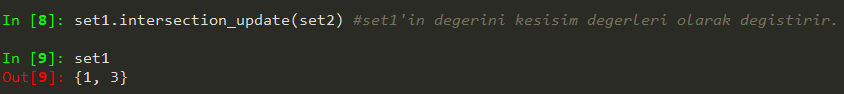
**intersection =** kesişim



**union =** birleşim



**intersection =** set1’in değerini kesişim değerleri olarak değiştirir.



#### Set’lerde Sorgu İşlemleri

**isdisjoint & issubset & issuperset**

**isdisjoint =** Ayrık küme mi?

İki kümenin kesişiminin boş olup olmadığını sorgular.

Boş ise True değil ise False döndürür.



**issubset =** subset’i mi? Alt kümesi mi? sorgusunu yapar.



**issuperset =** Kapsar mı?



### Veri Yapıları Özet

