In [1]:
```
# demet (tuple) tıpki liste yapısı gibi sıralı veri yapısıdır.
# sıralı veri yapılarının sahip olduğu ortak özelliklere sahiptir.
# Sıralıdır
# İndeks değerine sahiptir
# liste gibi tüm veri tiplerini içerisinde barındırabilir.
# listelerden farklı olarak değiştirilemeyen veri yapısıdır.
# değiştirmek gibi bir işlem yapılmaya çalışılırsa hata alınır.
```

In [2]:
```
# demet tanımında ( ) yapısı kullanılır
demet = ('Ali','Veli','Kırk Dokuz Elli')
demet
```

Out[2]:
```
('Ali', 'Veli', 'Kırk Dokuz Elli')
```

In [3]:
```
# boş liste tanımı
liste = list()
```

In [4]:
```
# boş demet yapısı oluşturma
demet = ()
demet
```

Out[4]:
```
()
```

In [5]:
```
# demetlerin tipi tuple
type(demet)
```

Out[5]:
```
tuple
```

In [6]:
```
# tuple olarak tanımlanmış kitap
kitap1=('Ahmed Arif','Hasretinden Prangalar Eskittim')
```

In [7]:
```
# tuple veri yapısı değiştirilemez veri yapısıdır ama
# tuple içerisinde değiştirilebilir veri yapılarını barındırabiliriz
demet = (['Python','Ruby','Perl'],['C++','C','C#'])
demet
```

Out[7]:
```
(['Python', 'Ruby', 'Perl'], ['C++', 'C', 'C#'])
```

In [8]:
```
# tuple ilk elemanı
demet[0]
```

Out[8]:
```
['Python', 'Ruby', 'Perl']
```

In [9]:
```
# tuple içerisinde bulunan listelerde değişiklik yapılabilir
# örneğin append ile tuple içerisindeki listeye ekleme yapabiliriz
demet[0].append('Lisp')
demet
```

Out[9]:
```
(['Python', 'Ruby', 'Perl', 'Lisp'], ['C++', 'C', 'C#'])
```

In [10]:
```
# tuple metodları sadece 2 metoda sahiptir
for i in dir(demet):
    if "_" not in i:
        print(i)
```

```
count
index
```

In [11]:
```
# tuple metodları sadece 2 metoda sahiptir
for i in dir(demet):
    print(i)
```

```
__add__
__class__
__class_getitem__
__contains__
__delattr__
__dir__
__doc__
__eq__
__format__
__ge__
__getattribute__
__getitem__
__getnewargs__
__gt__
__hash__
__init__
__init_subclass__
__iter__
__le__
__len__
__lt__
__mul__
__ne__
__new__
__reduce__
__reduce_ex__
__repr__
__rmul__
__setattr__
__sizeof__
__str__
__subclasshook__
count
index
```

In [12]:
```
# index() metodu tuple içerisinde
demet = ('A', 'B', 'C', 'D', 'E')
demet
```

Out[12]:
```
('A', 'B', 'C', 'D', 'E')
```

In [13]:
```
# 'C' elemanının indeks değeri
demet.index('C')
```

Out[13]:
```
2
```

In [14]:
```
# 'E' elemanının indeks değeri
demet.index('E')
```

Out[14]:
```
4
```

In [18]:
```
for i in demet:
    print(index(i),i)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [18], in <cell line: 1>()
      1 for i in demet:
----> 2     print(index(i),i)

NameError: name 'index' is not defined
```

In [19]:
```
len(demet)
```

Out[19]:
```
5
```

In [20]:
```
for i in range(len(demet)):
    print(i)
```

```
0
1
2
3
4
```

In [22]:
```
for i in range(len(demet)):
    print(i,demet[i],sep='-')
```

```
0-A
1-B
2-C
3-D
4-E
```

In [23]:
```
# count() metodu demet içerisindeki verinin kaç kez yer aldığını bulur
demet = (22,56,12,78,45,38,29,80,34,13,12,45,677,589,456,123,12)
demet
```

Out[23]:
```
(22, 56, 12, 78, 45, 38, 29, 80, 34, 13, 12, 45, 677, 589, 456, 123, 12)
```

In [24]:
```
# demet eleman sayısı
len(demet)
```

Out[24]:
```
17
```

In [25]:
```
# '12' elamanı demet içerisinde kaç defa geçiyor
demet.count(12)
```

Out[25]:
```
3
```

In [26]:
```
# '11' elamanı demet içerisinde kaç defa geçiyor
demet.count(11)
```

Out[26]:
```
0
```

In [27]:
```
languages = ('Python','C#','Java')type
languages
```

Out[27]:
```
('Python', 'C#', 'Java')
```

In [28]:
```
type(languages)
```

Out[28]:
```
tuple
```

In [29]:
```
# tuple alternatif kullanım
# parantez kullanmadan sadece virgül ile ayırarak elamanları yazabiliriz
diller = 'Python', 'C#', 'Java'
diller
```

Out[29]:
```
('Python', 'C#', 'Java')
```

In [30]:
```
type(diller)
```

Out[30]:
```
tuple
```

In [31]:
```
# boş tuple oluşturma
myTuple = ()
myTuple
```

Out[31]:
```
()
```

In [32]:
```
type(myTuple)
```

Out[32]:
```
tuple
```

In [33]:
```
# tek elemanlı tuple oluştururken dikkat etmek gerek
tup = ('Python')
tup
```

Out[33]:
```
'Python'
```

In [34]:
```
# tek elemanlı tuple oluşturmaya çalışırken str veri tipi döndü
type(tup)
```

Out[34]:
```
str
```

In [35]:
```
# tuple içerisindeki tek elemanın ardından , kullanılmalıdır
tup='Python',
tup
```

Out[35]:
```
('Python',)
```

In [36]:
```
type(tup)
```

Out[36]:
```
tuple
```

In [37]:
```
# tuple içerisindeki tek elemanın ardından , kullanılmalıdır
tup=('Python',)
tup
```

Out[37]:
```
('Python',)
```

In [38]:
```
type(tup)
```

Out[38]:
```
tuple
```

In [39]:
```
# len eleman sayısını verir
letters=('k','ı','z','a','k')
len(letters)
```

Out[39]:
```
5
```

In [40]:
```
# 'k' elemanı 2 defe geçiyor
letters.count('k')
```

Out[40]:
```
2
```

In [41]:
```
# tuple ilk elemana ulaşma
letters[0]
```

Out[41]:
```
'k'
```

In [42]:
```
# tuple son elemana ulaşma
letters[-1]
```

Out[42]:
```
'k'
```

In [43]:
```
# tuple sondan ikinci elemana ulaşma
letters[-2]
```

Out[43]:
```
'a'
```

In [44]:
```
# 'k' elemanının ilk bulduğu indeks değeri
letters.index('k')
```

Out[44]:
```
0
```

In [45]:
```
# 'z' elemanının ilk bulduğu indeks değeri
letters.index('z')
```

Out[45]:
```
2
```

In [46]:
```
# 1. index yani ikinci eleman
letters[1]
```

Out[46]:
```
'ı'
```

In [47]:
```
meyveler = ('elma','portakal','muz')
meyveler
```

Out[47]:
```
('elma', 'portakal', 'muz')
```

In [48]:
```
# tuple içerinde eleman değiştirilemez
# değiştirilen yapılar için list kullanımı tercih edilebilir
meyveler[1]='mandalina'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [48], in <cell line: 2>()
      1 # tuple içerinde eleman değiştirilemez
----> 2 meyveler[1]='mandalina'

TypeError: 'tuple' object does not support item assignment
```

In [49]:
```
sebzeler = ('Domates','Biber','Patlıcan')
sebzeler
```

Out[49]:
```
('Domates', 'Biber', 'Patlıcan')
```

In [50]:
```
# tuple birleştirme
meyve_sebze = meyveler + sebzeler
meyve_sebze
```

Out[50]:
```
('elma', 'portakal', 'muz', 'Domates', 'Biber', 'Patlıcan')
```

In [51]:
```
# tuple içerisinde benersiz kaç eleman bulunur
sayilar = (1,3,5,6,8,3,8)
len(sayilar)
```

Out[51]:
```
7
```

In [52]:
```
# set() ile küme veri tipine dönüştürülür
# küme tipinde her eleman yalnızca birkez kullanılır
# küme veri tipi {1, 3, 5, 6, 8} gibi tanımlanır
kume = set(sayilar)
kume
```

Out[52]:
```
{1, 3, 5, 6, 8}
```

In [53]:
```
# kümenin eleman sayısı
len(kume)
```

Out[53]:
```
5
```

In [54]:
```
sayilar = (1,3,5,6,8,3,8)
len(set(sayilar))
```

Out[54]:
```
5
```

In [55]:
```
# benzersiz eleman sayısını bulan fonksiyon
def benzersiz_eleman_sayisi(tuple):
    return len(set(tuple))
```

In [56]:
```
sayilar = (1,3,5,6,8,3,8,2,6,4,1)
benzersiz_eleman_sayisi(sayilar)
```

Out[56]:
```
7
```

In [57]:
```
# benzersiz elemanları gösteren fonksiyon
def benzersiz_eleman_goster(tuple):
    return set(tuple)
```

In [58]:
```
benzersiz_eleman_goster(sayilar)
```

Out[58]:
```
{1, 2, 3, 4, 5, 6, 8}
```

In [59]:
```
sayilar
```

Out[59]:
```
(1, 3, 5, 6, 8, 3, 8, 2, 6, 4, 1)
```

In [60]:
```
# list() fonk. ile tuple  veri tipi liste tipine dönüştürülür
sayi_listem = list(sayilar)
sayi_listem
```

Out[60]:
```
[1, 3, 5, 6, 8, 3, 8, 2, 6, 4, 1]
```

In [61]:
```
type(sayi_listem)
```

Out[61]:
```
list
```

In [62]:
```
# tuple içindeki elemanları tek bir stringde toplama
# ayraç olarak kullanacağımız metin ',' üzerinden
# join() fonk. parametre olarak tuple verilir

people = ('Hakan','Ercan','Melih','Osman')
text = ','.join(people)
text
```

Out[62]:
```
'Hakan,Ercan,Melih,Osman'
```

In [63]:
```
type(text)
```

Out[63]:
```
str
```

In [64]:
```
# tuple içindeki değerleri ayrı ayrı değişkenlere atama
values = (7, 13)
values
```

Out[64]:
```
(7, 13)
```

In [65]:
```
# aralarında ',' kullanarak değişkenleri tuple eşitleriz
value1 , value2 = values
value1 , value2
# Python elemanların çözümlemelerini kendisi yaparak değişkenlere atar
```

Out[65]:
```
(7, 13)
```