**BÖLÜM 11**

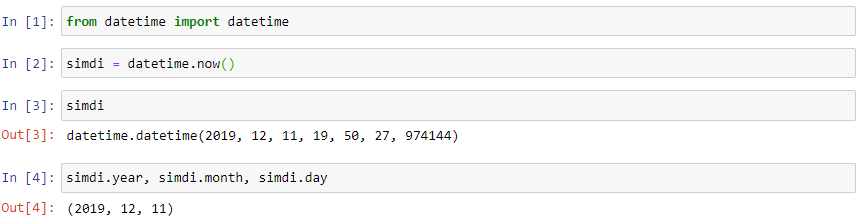
**ZAMAN SERİSİ**

Zaman serisi verileri, finans, ekonomi, ekoloji, sinirbilim ve fizik gibi birçok farklı alanda yapılandırılmış önemli bir veri biçimidir. Zaman içinde birçok noktada gözlemlenen veya ölçülen herhangi bir şey bir zaman serisi oluşturur. Çoğu zaman serisi sabit frekanstır, yani veri noktalarının her 15 saniyede bir, 5 dakikada bir veya ayda bir kez olduğu gibi bazı kurallara göre düzenli aralıklarla gerçekleştiği söylenebilir. Zaman serileri ayrıca sabit bir zaman birimi olmadan düzensiz olabilir veya birimler arasında dengelenebilir.

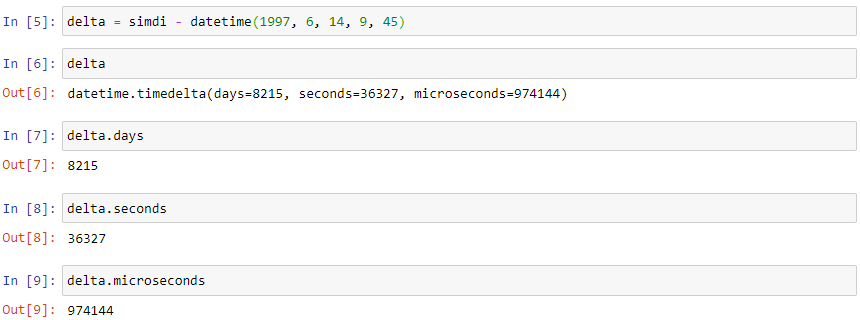
Pandas birçok yerleşik zaman serisi aracı ve veri algoritması sağlar. Çok büyük zaman serileri ile verimli bir şekilde çalışabilir, düzensiz ve sabit frekanslı zaman serilerini kolayca dilimleyebilir, kümeleyebilir ve yeniden örnekleyebilirsiniz.

**11.1. Tarih ve Saat Veri Türleri ve Araçları**

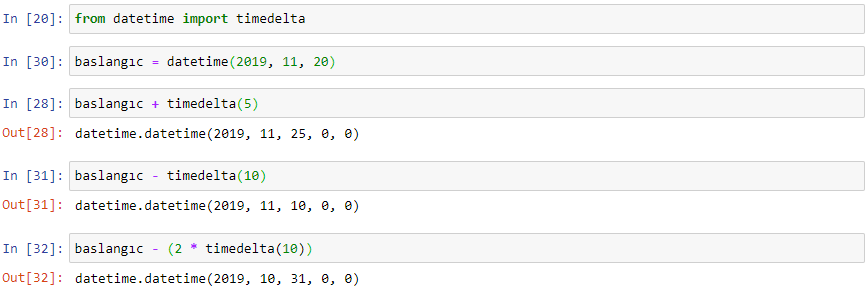
Python standart kütüphanesi, tarih ve saat verileri için veri tiplerinin yanı sıra takvimle ilgili fonksiyonlarda içerir. Tarih, saat ve takvim modülleri başlangıç için temel yerlerdir. *Datetime.datetime* türü veya yalnızca *datetime* yaygın olarak kullanılır.



*datetime*, tarih ve saati mikro saniyeye indirir. *timedelta*, iki *datetime* nesnesi arasındaki zamansal farkı temsil eder.



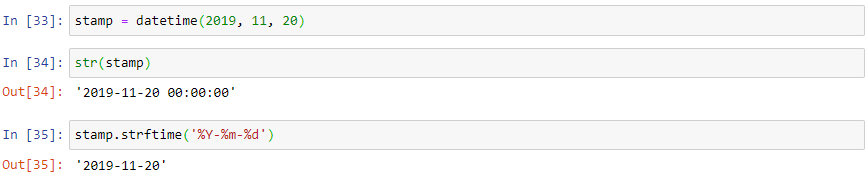
Yeni bir kaydırılmış nesne üretmek için *datetime* nesnesine bir *timedelta* veya bunun birçoğunu ekleyebilirsiniz.



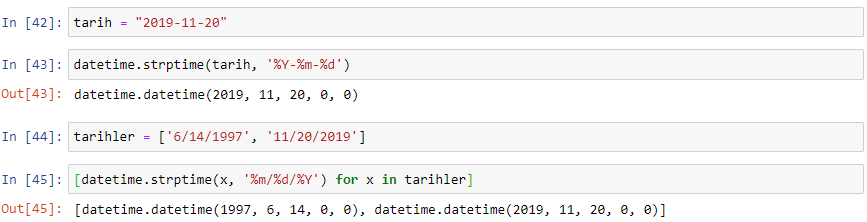
Tablo: datetime modülündeki veri türlerini özetlemektedir.

|  |  |
| --- | --- |
| **Tip** | **Açıklama** |
| date | Gregoryen takvimini kullanarak takvim tarihini (yıl, ay, gün) depolar |
| time | Günün saatini saat, dakika, saniye ve mikro saniye olarak depolar |
| datetime | Hem tarih hem de saat depolar |
| timedelta | İki *datetime* değeri arasındaki farkı temsil eder (gün, saniye ve mikrosaniye gibi) |
| tzinfo | Saat dilimi bilgilerini depolamak için temel tür |

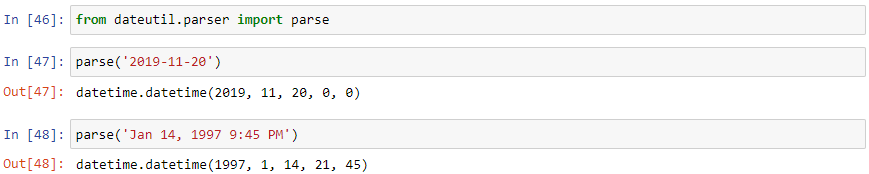
**String ve Datetime Arasında Dönüşüm**



*Datetime.strptime* kullanarak stringleri tarihe dönüştürebiliriz.



*datetime.strptime* bilinen biçimde bir tarihi ayrıştırmak için iyi bir yoldur. Bununla birlikte, özellikle genel tarih biçimleri için her defasında bir biçim belirtimi yazmak biraz can sıkıcı olabilir. Bu durumda, parser.parse yöntemini üçüncü taraf dateutil paketini kullanabiliriz.



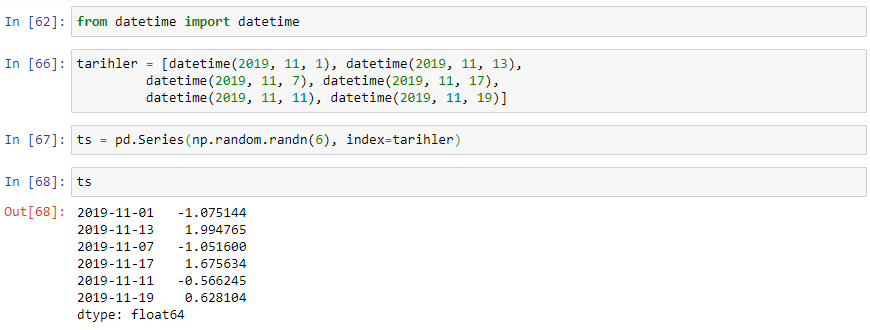
Uluslararası yerel bölgelerde aydan önce görünen gün çok yaygındır. Bu nedenle dayfirst = True komutu ile bu ayarı yapabiliriz.



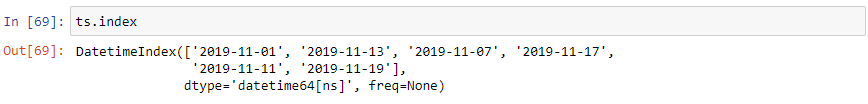


**11.2. Zaman Serisi Temelleri**

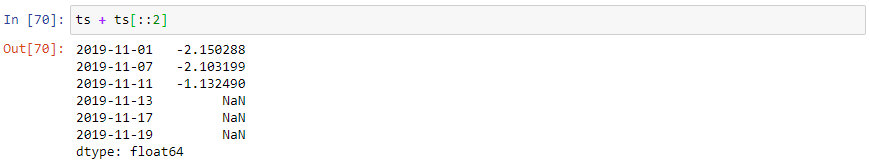
Pandas temel bir zaman serisi nesnesi, genellikle pandas dışında Python stringleri veya datetime nesneleri olarak temsil edilen, timestamps tarafından indekslenen bir seridir.



*datetime* nesneleri bir *DatetimeIndex* öğesine yerleştirilir.



Diğer serilerde olduğu gibi, farklı indekslenmiş zaman serileri arasındaki aritmetik işlemler otomatik olarak tarihlere göre sıralanır.



Pandas, nanosaniye çözünürlükte NumPy’nin datetime64 veri türünü kullanarak zaman damgalarını depolar.

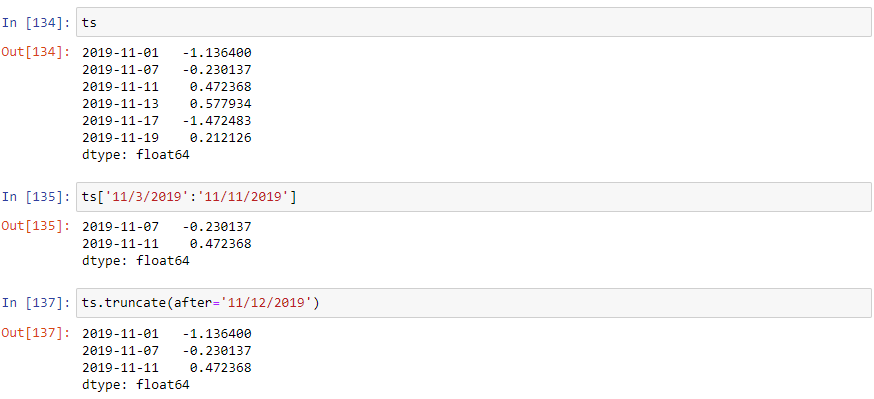
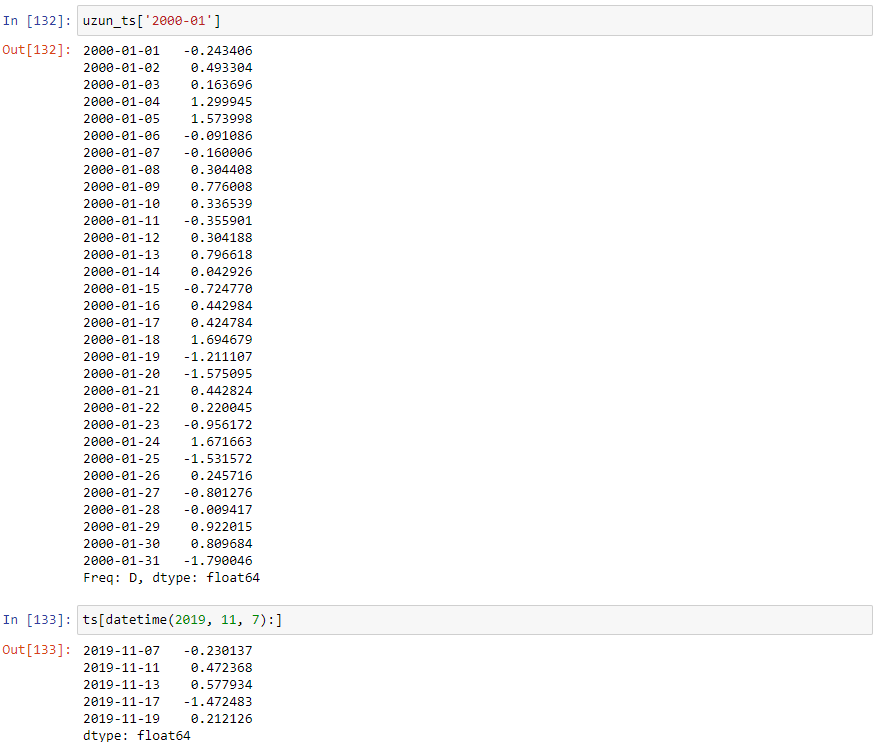
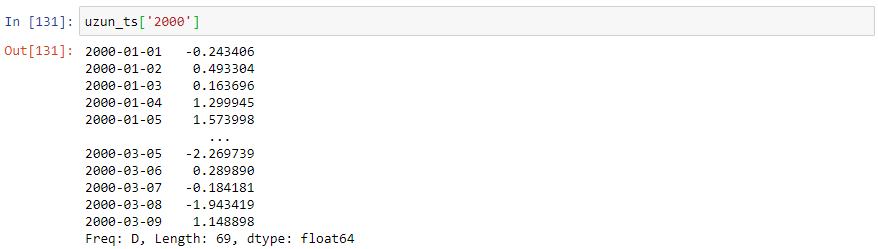


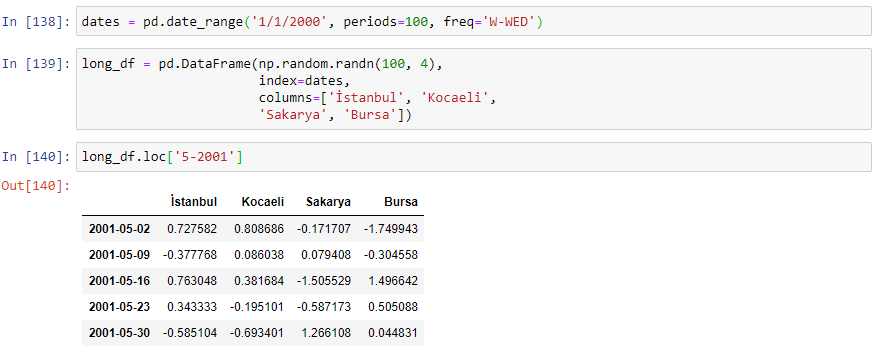
Bir datetimeindex öğesinden skaler değerler timestamp nesneleridir.



**İndeksleme, Seçim, Alt Kümeleme**

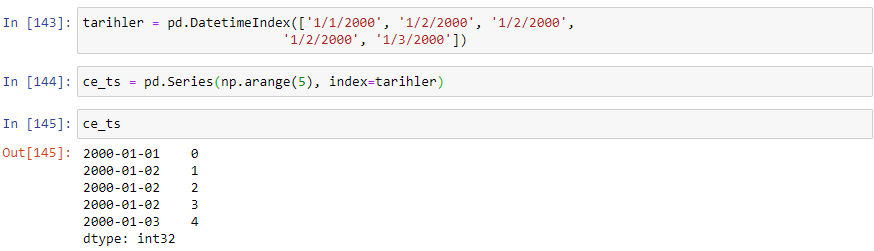






**Çift Endeksli Zaman Serileri**

Bazı uygulamalarda, belirli bir zaman damgasına düşen çoklu veri gözlemleri olabilir:



is\_unique özelliği ile diziyi kontrol ederek benzersiz olup olmadığını kontrol edebiliriz.



Bu zaman sersinde zaman damgasının kopyalanıp kopyalanmadığına bağlı olarak skaler değerler veya dilimler döndürecektir.



Benzersiz olmayan zaman damgalarına sahip verileri toplamak istediğimizde groupby kullanabiliriz. (level = 0)



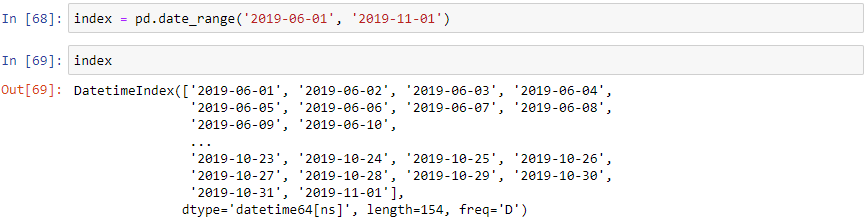
**11.3. Tarih Aralıkları, Frekanslar ve Kaydırma**



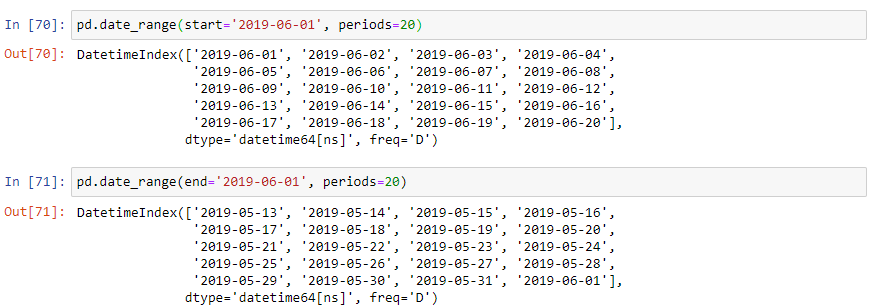
'D' stringi günlük frekans olarak yorumlanır.

**Tarih Aralıkları Oluşturma**

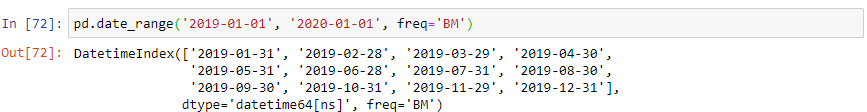
*pandas.date\_range*, belirli bir frekansa göre belirtilen uzunlukta bir *datetimeindex* oluşturur.



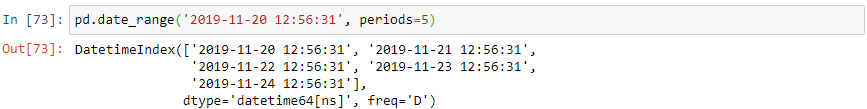
Varsayılan olarak, *date\_range* günlük zaman damgaları oluşturur. Yalnızca bir başlangıç veya bitiş tarihi ve *periods* değişkeni ile kaç dönem ilerleneceği belirtilebilir.



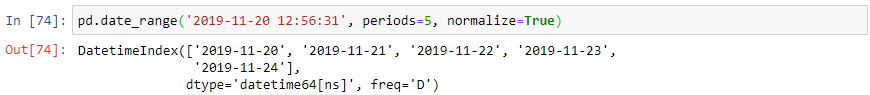
Başlangıç ve bitiş tarihleri, oluşturulan tarih dizini için kesin sınırları tanımlar. Örneğin, her ayın son iş gününü içeren bir tarih dizini istiyorsak 'BM' periyodunu kullanabiliriz.



*date\_range* varsayılan olarak, başlangıç veya bitiş zaman damgasının zamanını (varsa) korur.

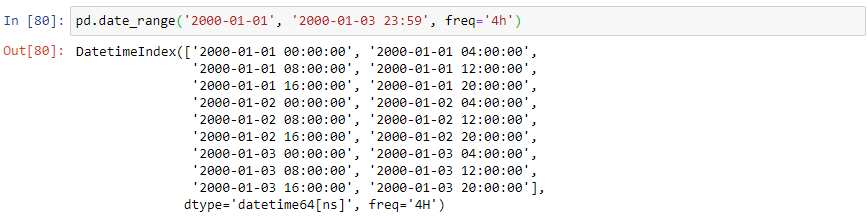
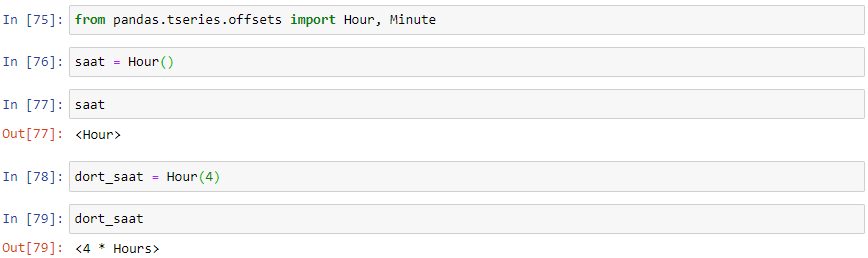


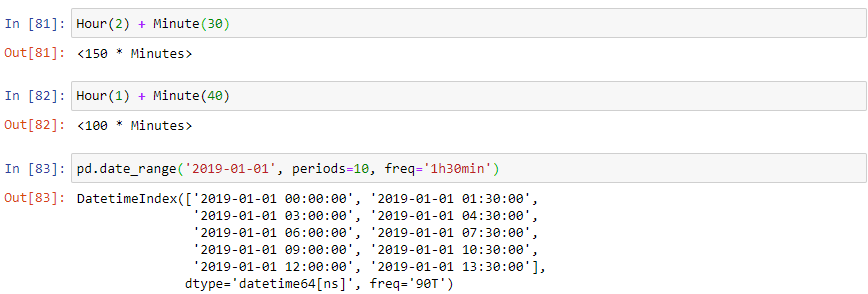
Bazen tarihlerde zaman bilgisi aynı olabilir, ancak normalize edilmiş bir zaman damgası kümesi oluşturmak isteyebilirsiniz. Bunu yapmak için normalleştirme seçeneği vardır.



**Frekanslar ve Tarih Ofsetleri**

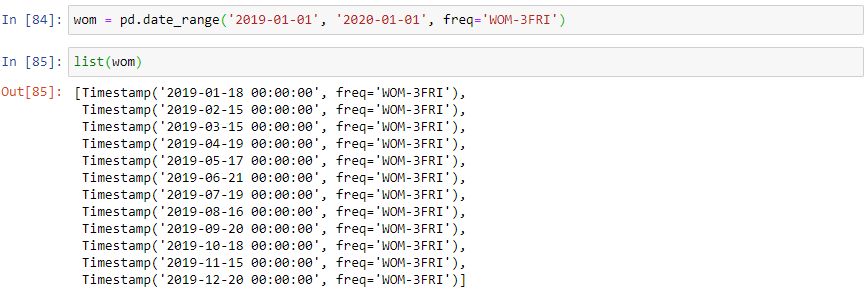
Pandas’da, frekanslar bir baz frekans ve bir çarpandan oluşur. Temel frekanslara tipik olarak, aylık olarak 'M' veya saatlik olarak 'H' gibi bir string takma adı verilir. Her bir baz frekans için, genellikle tarih ofseti olarak adlandırılan bir nesne vardır. Örneğin, saat frekansı Hour sınıfı ile temsil edilir.





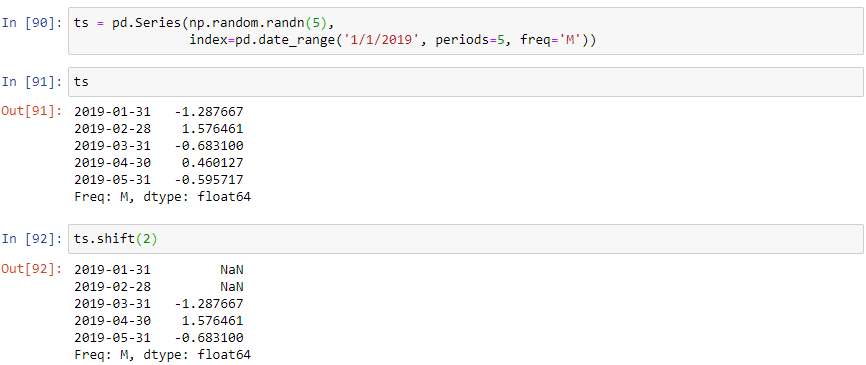
**Ayın Haftası**

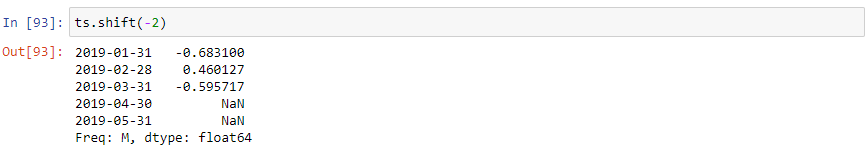
WOM (Week of Month) ile bize “Ayın haftası" bilgisini döndürür. Bu, her ayın üçüncü Cuma günü gibi tarihleri almanızı sağlar.

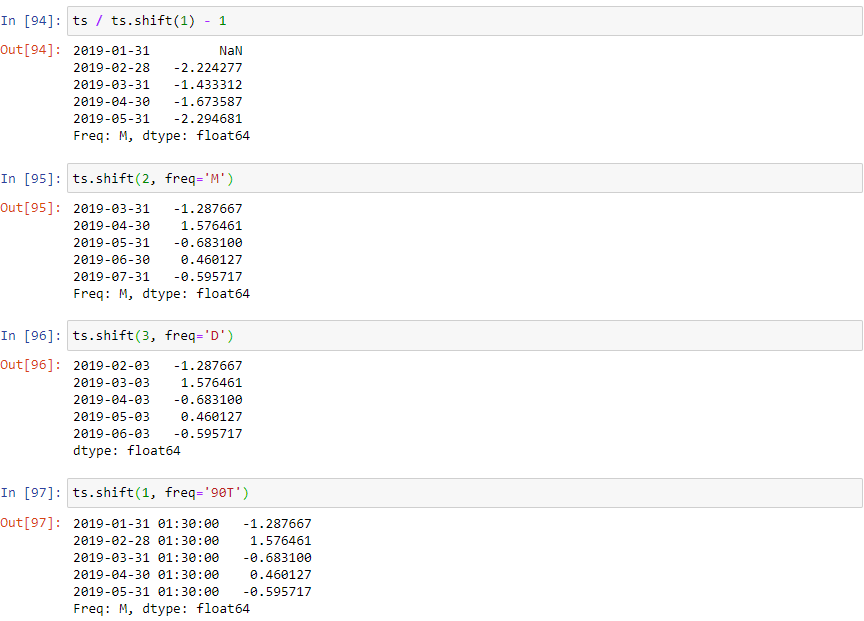


**Kaydırma (Öncü ve Gecikmeli) Veri**

“Kaydırma”, verileri zaman içinde geriye ve ileriye doğru hareket ettirmeyi ifade eder. Hem Seriler hem de DataFrame, saf kaydırma yapmak için ileri ya da geri kaydırırken, dizini değiştirilmemiş halde bırakan bir yönteme sahiptir.





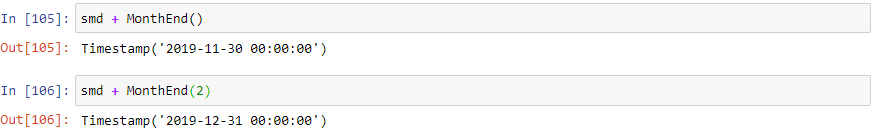


**Tarihleri Ofsetlerle Değiştirme**

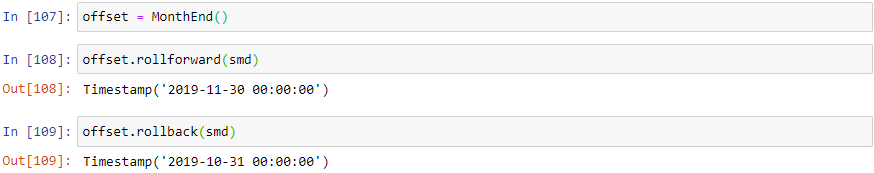
Pandas tarih ofsetleri *datetime* veya *timestamp* nesneleriyle de kullanılabilir.

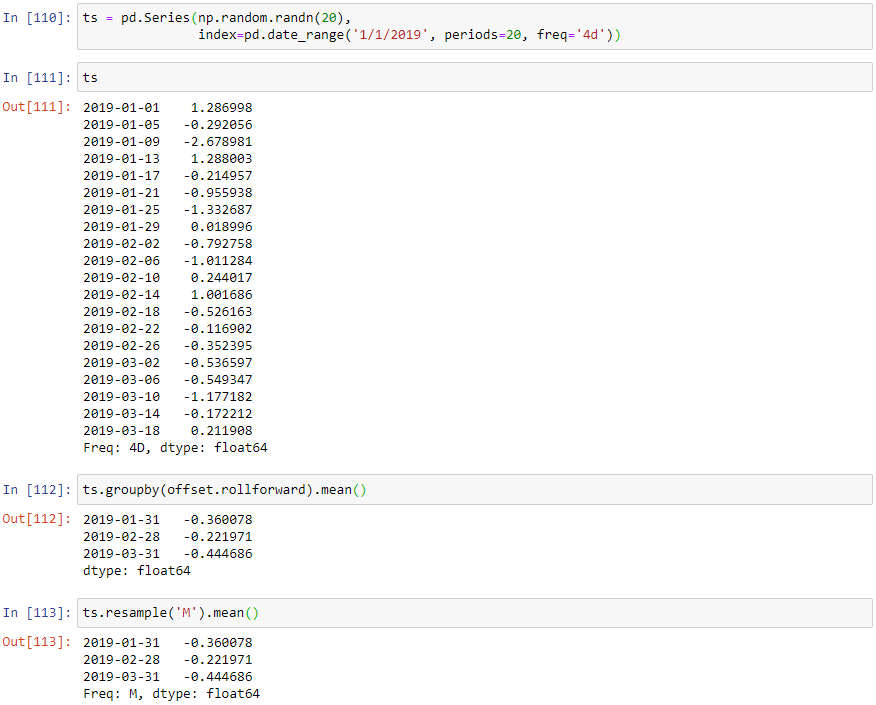


MonthEnd gibi bağlantılı bir ofset eklerseniz, artış kuralına göre ayın son tarihine götürür.



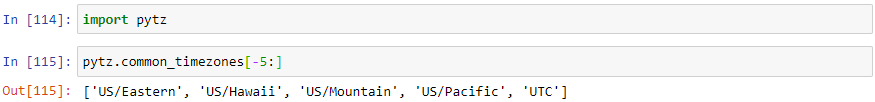
Sırasıyla *rollforward* ve *rollback* yöntemlerini kullanarak tarihi ileri veya geri taşıyabiliriz.





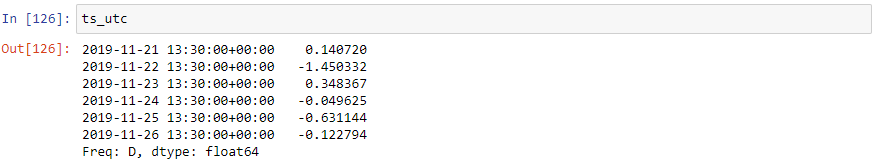
**11.4. Saat Dilimi Kullanımı**

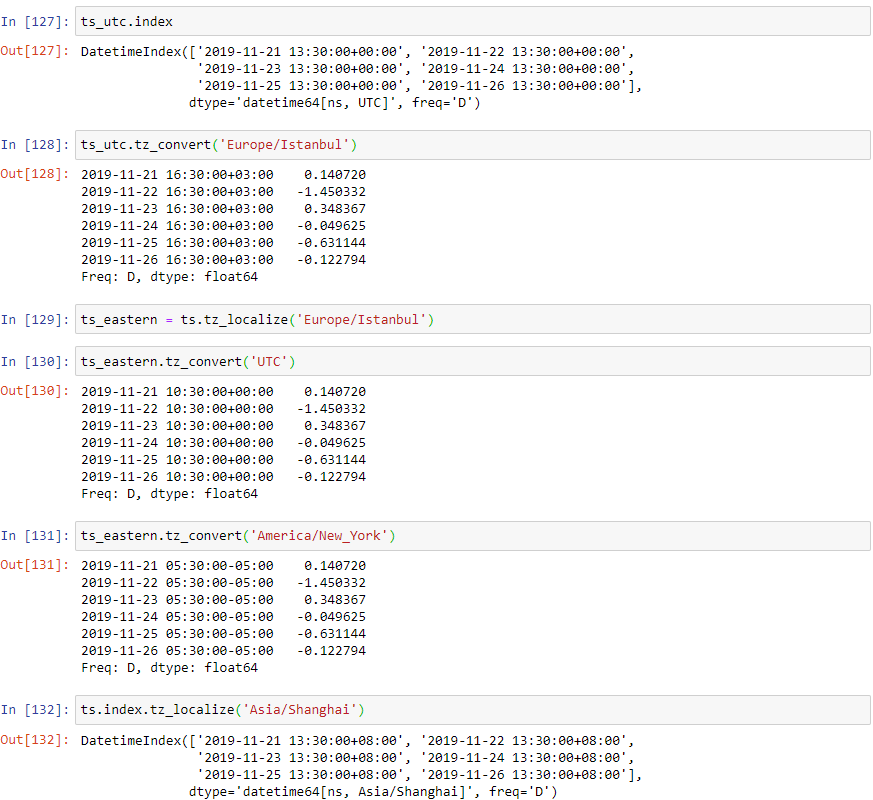
Zaman dilimleri ile çalışmak genellikle zaman serileri manipülasyonunun en sıkıcı parçalarından biri olarak kabul edilir. Sonuç olarak, birçok zaman serisi kullanıcısı, zaman çizelgesi ile koordineli bir evrensel zaman veya Greenwich Ortalama Zamanının devamı olan UTC’de çalışmayı tercih eder ve şu anki uluslararası standarttır. Python'da saat dilimi bilgisi, dünya saat dilimi bilgisinin bir derlemesi olan Olson veri tabanını ortaya koyan üçüncü taraf pytz kütüphanesinden gelir. Bu özellikle tarihsel veriler için önemlidir.





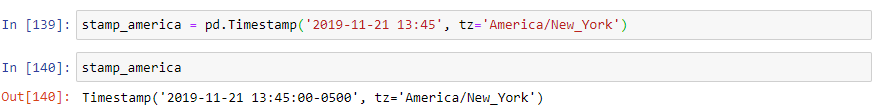
**Saat Dilimi Yerelleştirme ve Dönüşüm**

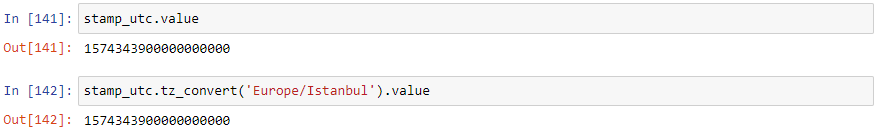


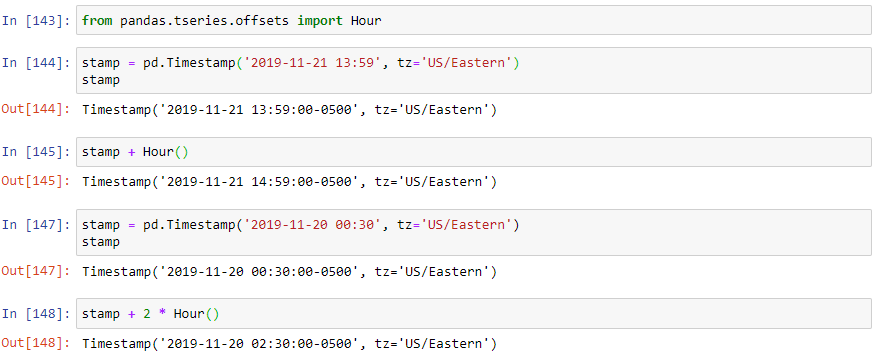


**Saat Dilimi ile İşlemler − Bilinen Zaman Damgası Nesneleri**

Zaman serileri ve tarih aralıklarına benzer şekilde, ayrı ayrı *timestamp* nesneleri benzer şekilde saf saat dilimine göre yerelleştirilebilir ve bir saat diliminden diğerine dönüştürülebilir.







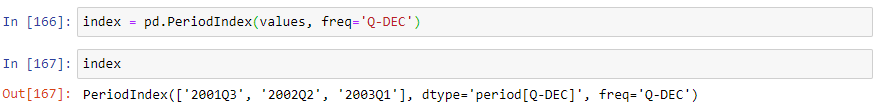
**Farklı Zaman Dilimleri Arasındaki İşlemler**

Farklı zaman dilimleri ile iki zaman serisi birleştirilirse, sonuç UTC olacaktır. Zaman damgaları UTC'de saklandığından, bu basit bir işlemdir ve herhangi bir dönüşüm yapılmasını gerektirmez.



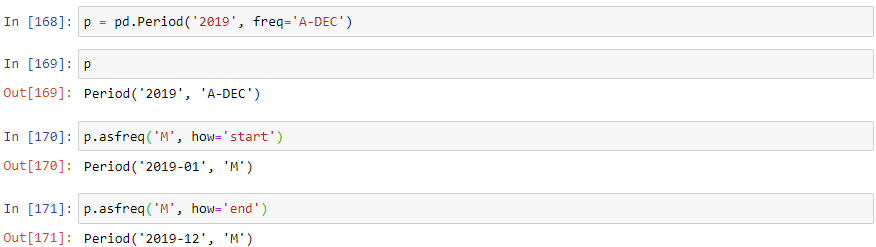
**11.5. Dönemler ve Dönem Aritmetiği**

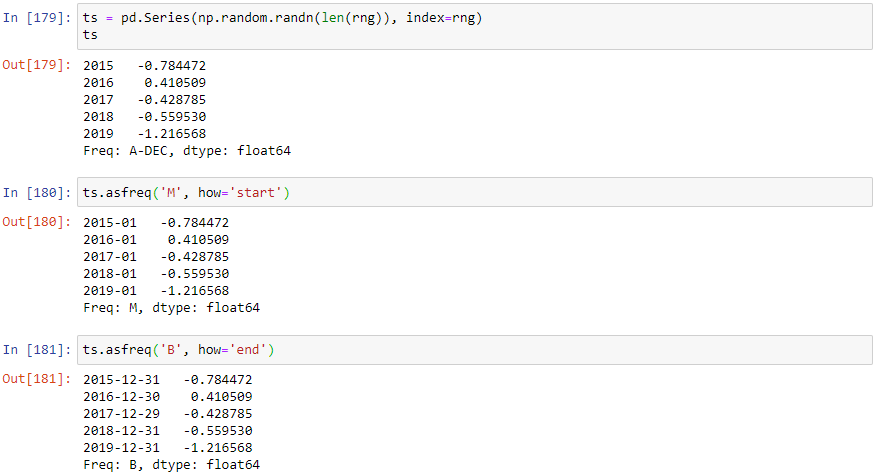
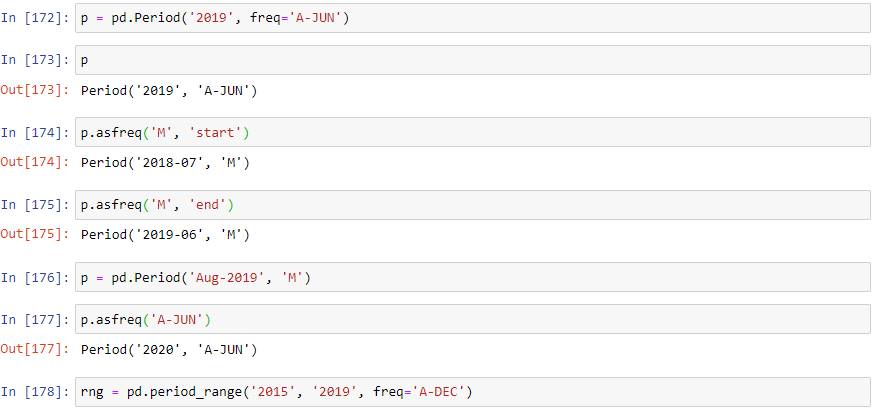
Dönemler, günler, aylar, çeyrekler veya yıllar gibi zaman aralıklarını temsil eder.



**Periyot Frekans Dönüşümü**

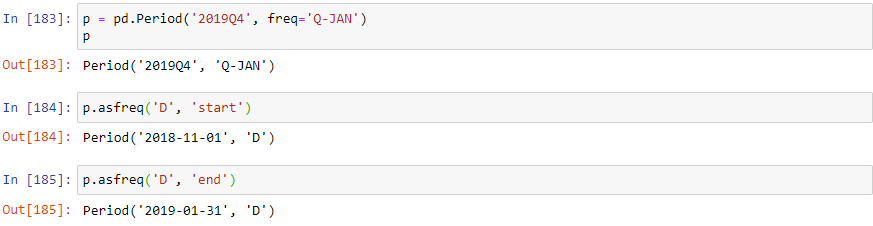
Dönemler ve *PeriodIndex* nesneleri, *asfreq* yöntemleriyle başka bir frekansa dönüştürülebilir. Örnek olarak, yıllık bir dönemimiz olduğunu ve bunu yılın başında veya sonunda bir aylık döneme dönüştürmek istediğimizi varsayalım.

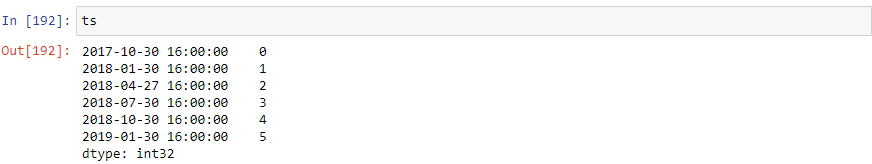




**Üç Aylık Dönem Frekansları**

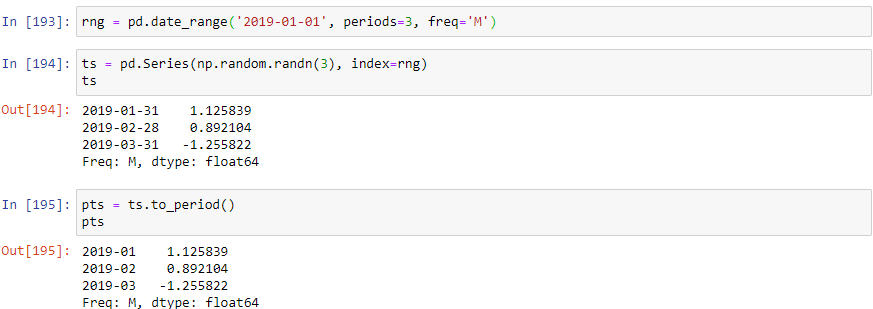
Üç ayda bir veri muhasebe, finans ve diğer alanlarda standarttır. Üç aylık veri, mali yılın sonuna göre, tipik olarak yılın 12 ayından birinin son takvimi veya iş günü olarak rapor edilir. Dolayısıyla, 2012Ç4 dönemi mali yılsonuna bağlı olarak farklı bir anlama sahiptir. Pandas, Q-JAN - Q-DEC'e kadar olan üç olası tüm frekansları destekler.



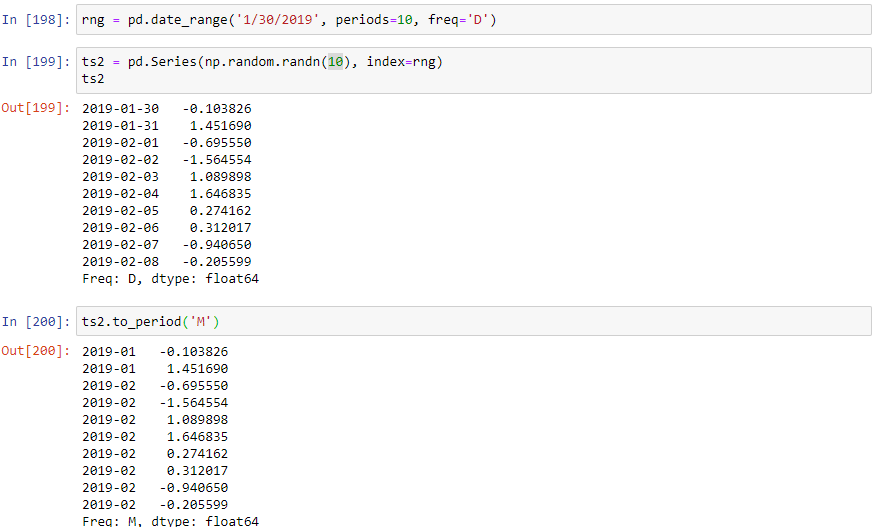


**Zaman Damgalarını Döneme (ve geri) Dönüştürme**

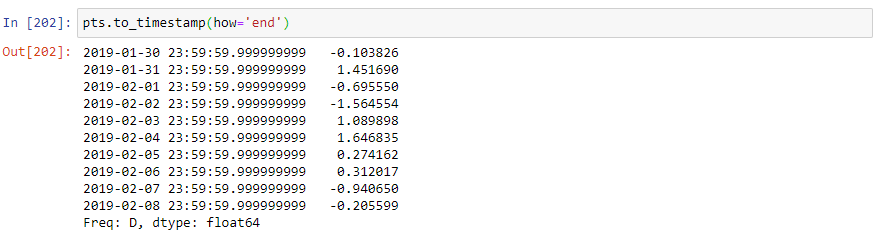
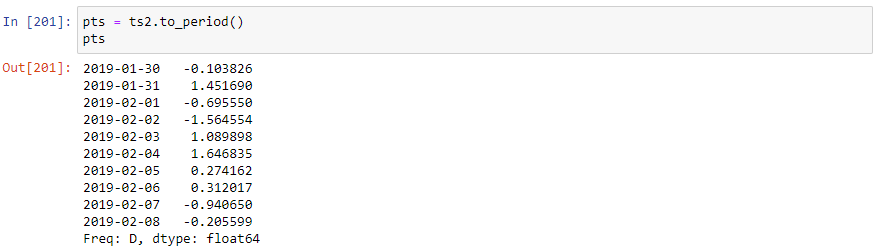
Zaman damgaları tarafından indekslenen Series ve DataFrame nesneleri, *to\_period* yöntemiyle dönemlere dönüştürülebilir.



Dönemler çakışmayan zaman aralıklarına işaret ettiğinden, bir zaman damgası yalnızca belirli bir frekans için tek bir döneme ait olabilir. Yeni dönem dizininin sıklığı varsayılan olarak zaman damgalarından çıkarılırken, istediğiniz frekansı belirtebilirsiniz. Sonuç olarak yinelenen dönemlere sahip olmak da sorun değildir.

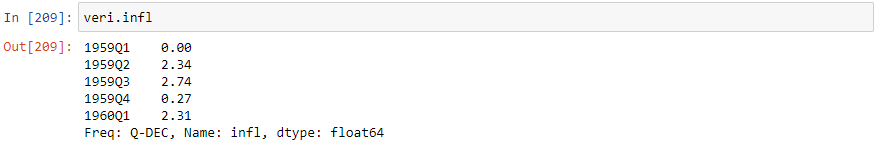
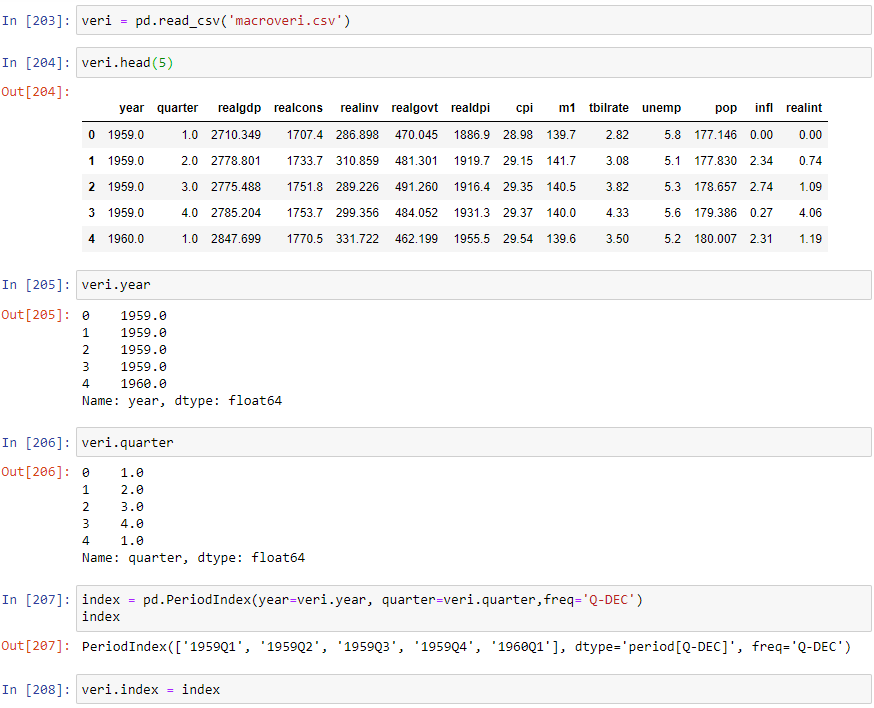


Zaman damgalarına geri dönmek için, *to\_timestamp* kullanılır.



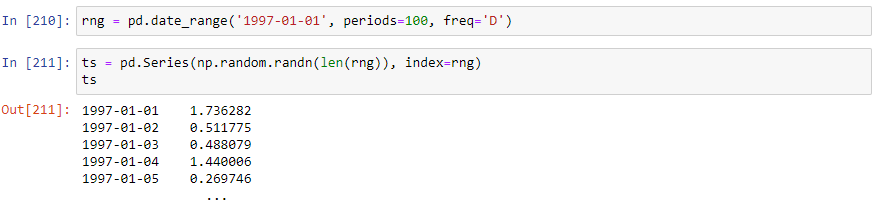
**Dizilerden PeriodIndex Oluşturma**

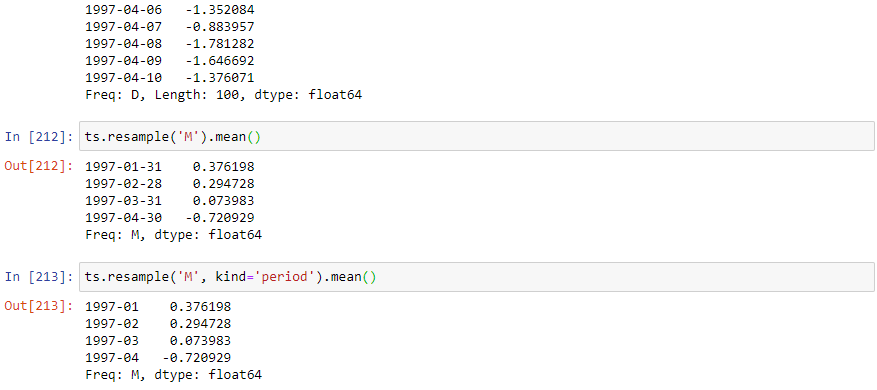
Sabit frekanslı veri kümeleri, bazen birden çok sütuna yayılmış zaman dilimi bilgileriyle saklanır. Örneğin, bu makroekonomik veri setinde, yıl ve çeyrek farklı sütunlardadır.



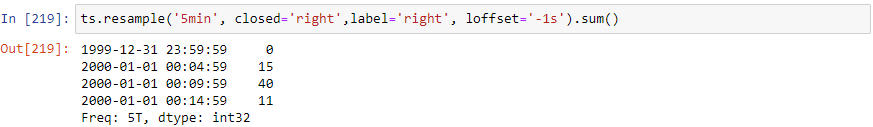
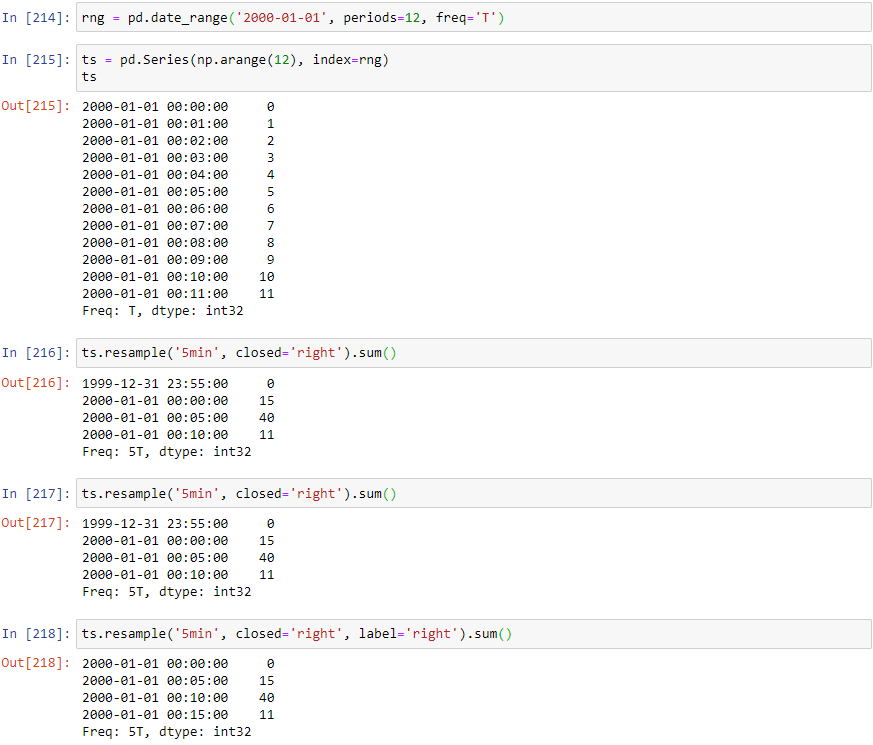
**11.6. Yeniden Örnekleme ve Frekans Dönüştürme**

Yeniden örnekleme, bir zaman serisini bir frekanstan diğerine dönüştürme işlemini ifade eder. Daha yüksek frekans verilerini daha düşük frekansa toplamak, daha düşük frekansı daha yüksek frekansa dönüştürmek için numune almaya denir. Pandas nesneleri, tüm frekans dönüşümleri için iş gücü işlevi olan bir resample yöntemiyle donatılmıştır. Resample, groupby ile benzer bir API'ye sahiptir; veriyi gruplandırmak için resample öğesini çağırır, ardından bir toplama işlevini çağırırsınız.



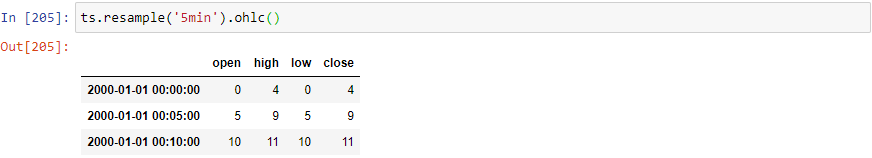


**Alt Örnekleme**



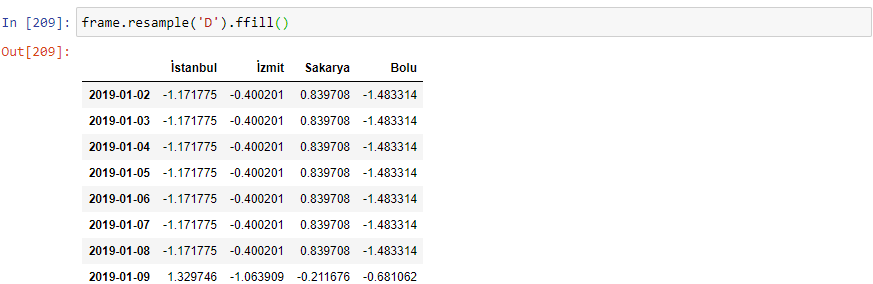
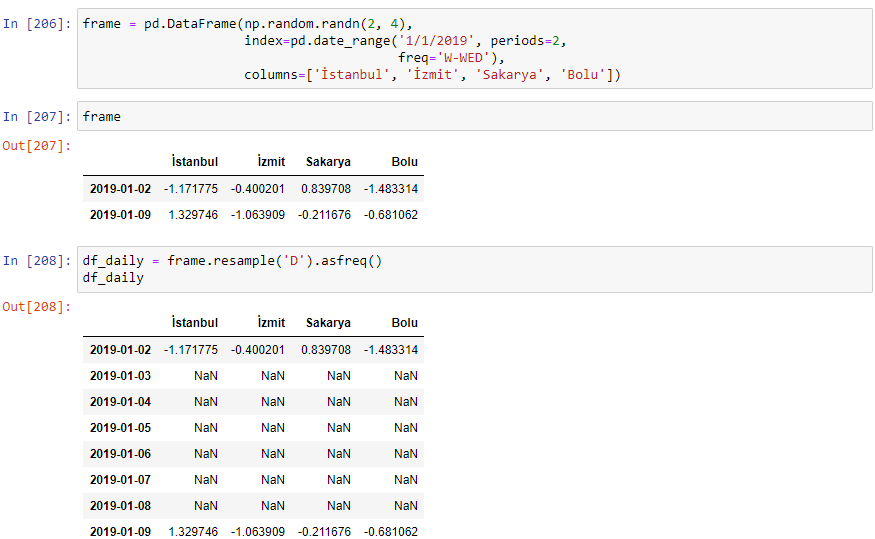
**Açık-Yüksek-Düşük-Kapalı (OHLC) Örneklemesi**

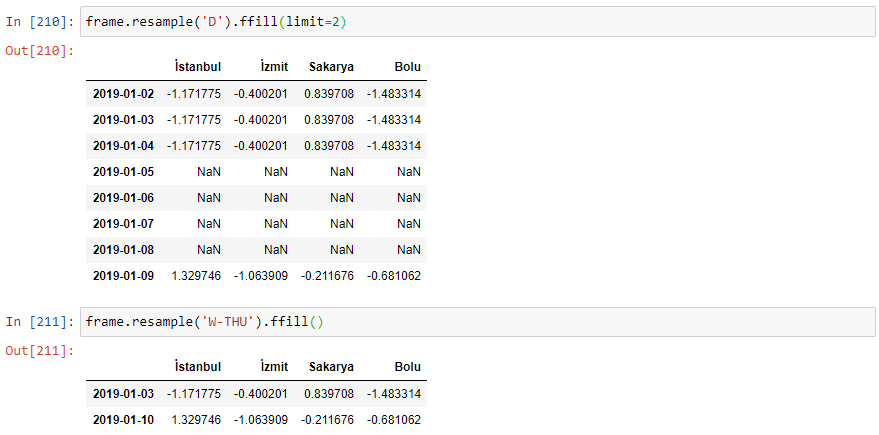
Finansta, bir zaman serisini bir araya getirmenin popüler bir yolu, her bir kova için dört değer hesaplamaktır: ilk (açık), son (kapalı), maksimum (yüksek) ve minimum (düşük) değerler. OHLC toplama işlevini kullanarak, bu dört kümeyi içeren sütunlara sahip bir DataFrame elde edilir.



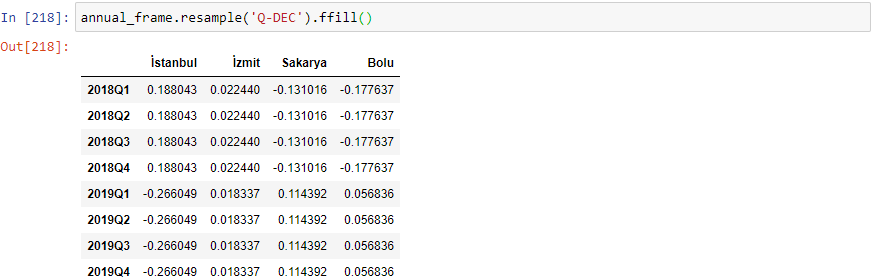
**Örnekleme ve İnterpolasyon**

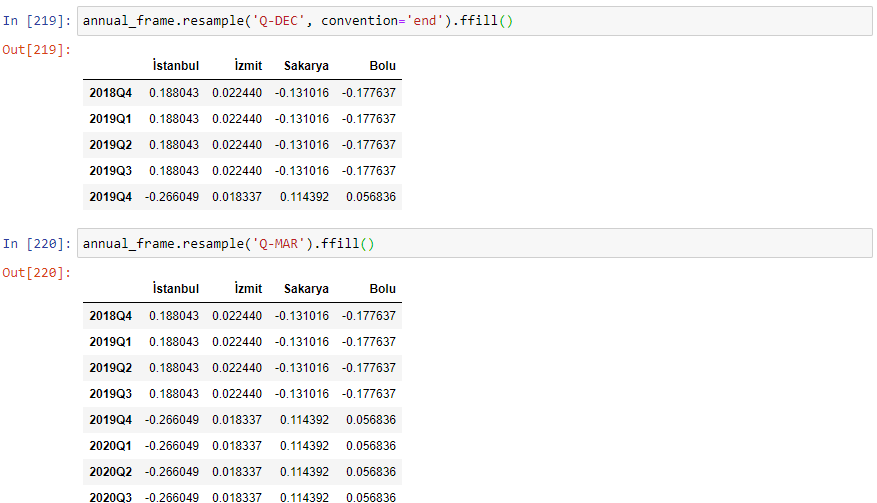
Düşük bir frekanstan daha yüksek bir frekansa dönüştürürken, toplama gerekmez.



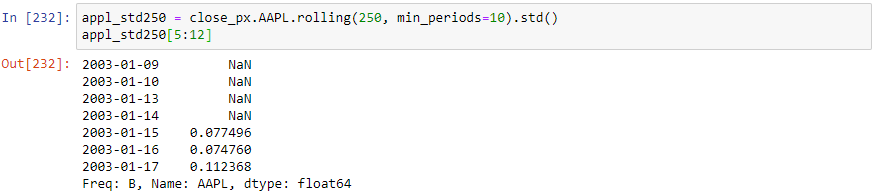
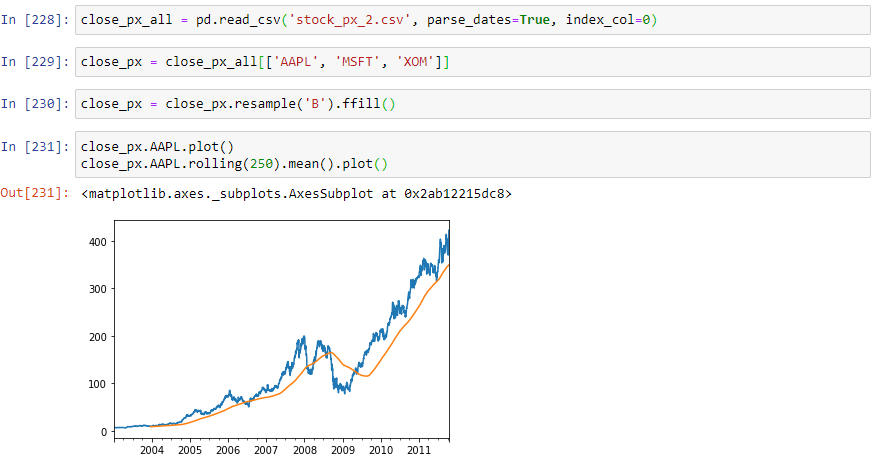


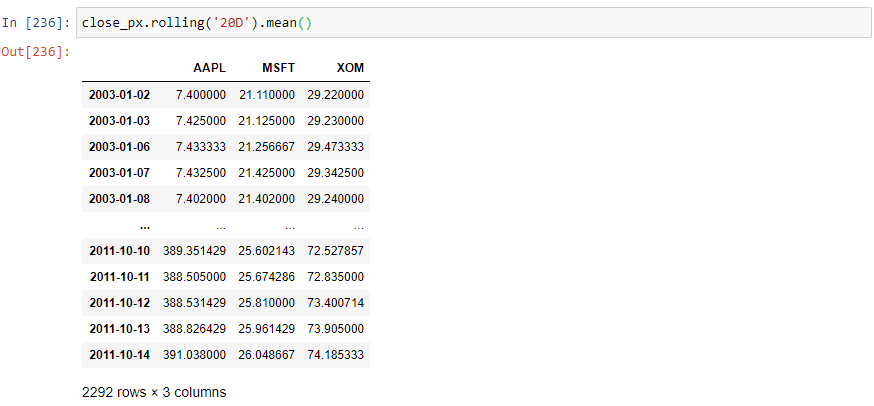
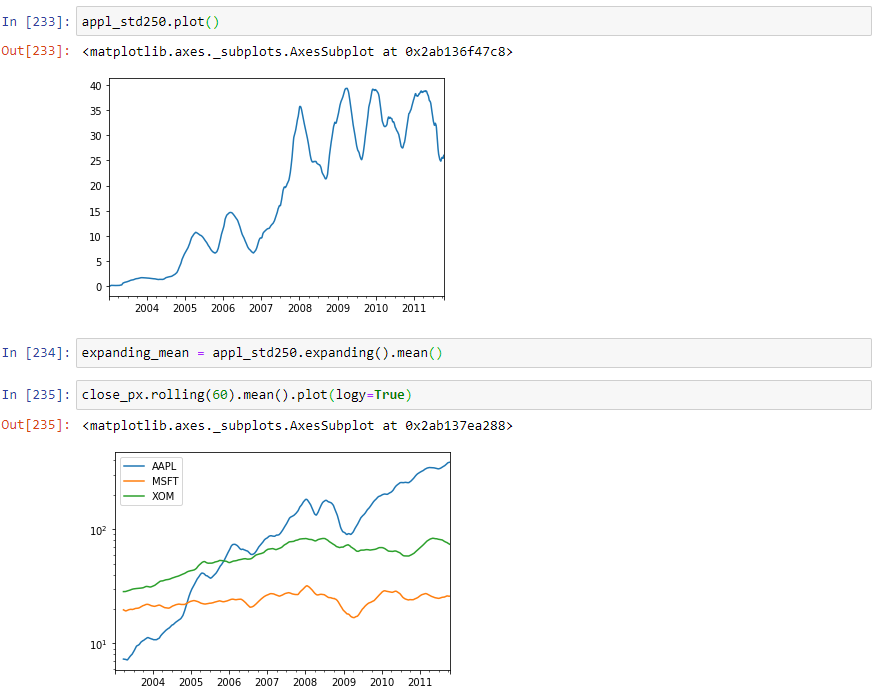
**Dönemlerle Yeniden Örnekleme**





**11.7. Hareketli Pencere Fonksiyonları**





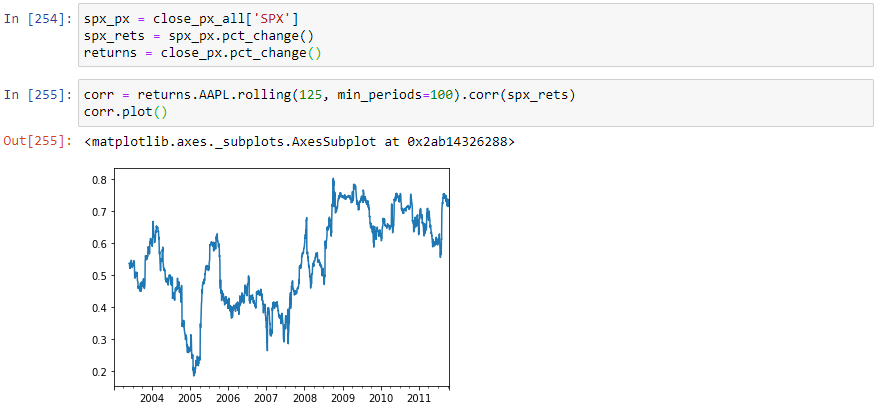
**Üstel Ağırlıklı İşlevler**

Eşit ağırlıklı gözlemlerle statik pencere boyutu kullanmanın bir alternatifi, daha yeni gözlemlere daha fazla ağırlık vermek için sabit bir bozulma faktörü belirlemektir. Bozulma faktörünü belirtmenin birkaç yolu vardır. Popüler olanı bir yayılma alanı kullanmaktır; bu da sonucu, yayılma alanına eşit pencere boyutuyla basit bir hareketli pencere işlevi ile karşılaştırılabilir kılar. Üstel ağırlıklı bir istatistik daha son gözlemlere daha fazla ağırlık verdiğinden, eşit ağırlıklı versiyona kıyasla değişikliklere daha hızlı adapte olur.

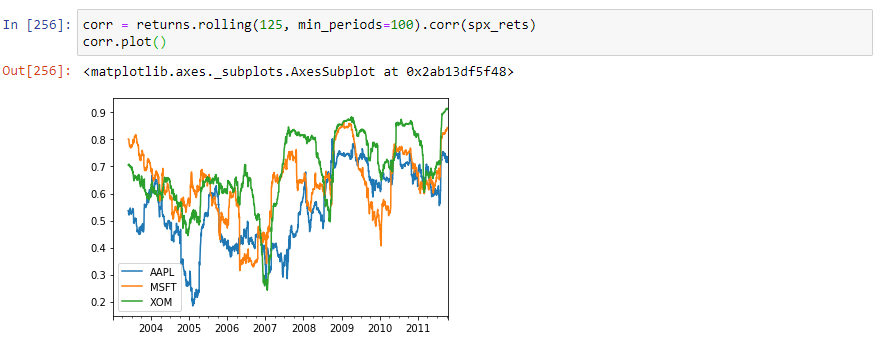


**İkili Hareketli Pencere Fonksiyonları**

Korelasyon ve kovaryans gibi bazı istatistiksel operatörlerin iki zaman serisinde çalışması gerekir. Örnek olarak, finansal analistler genellikle bir hisse senedinin S&P 500 gibi bir gösterge endeks ile korelasyonuyla ilgileniyorlar. Buna bir göz atmak için, öncelikle tüm ilgi alanımızdaki yüzde değişimini hesaplanır.

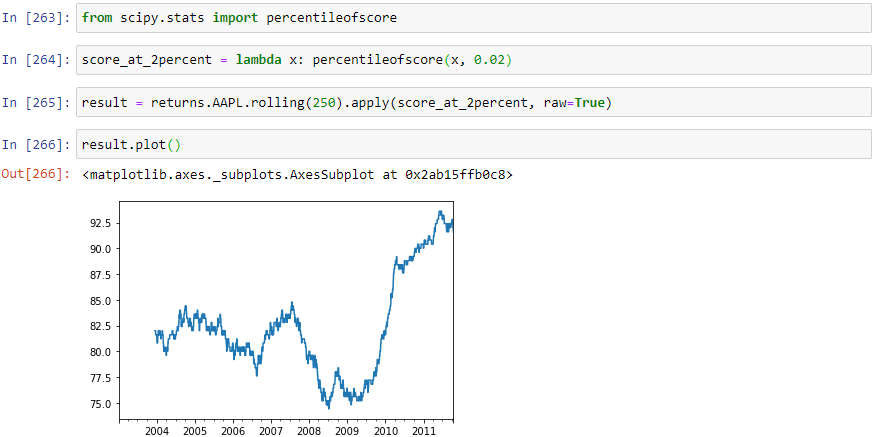


S&P 500 endeksinin korelasyonunu bir seferde birçok stokla hesaplamak istediğimizi varsayalım. Bir döngü yazmak ve yeni bir DataFrame oluşturmak kolay olabilir ancak tekrarlanabilir hale gelir, bu nedenle bir Seri ve DataFrame iletilirse, *rolling\_corr* gibi bir işlev Series'in (spx\_rets) DataFrame içindeki her sütunla korelasyonunu hesaplar.



**Kullanıcı Tanımlı Hareketli Pencere Fonksiyonları**

*rolling* ve ilgili yöntemlerde *apply* yöntemi, kendi geliştirdiğiniz hareketli bir pencerede bir dizi işlevi uygulamak için bir yol sağlar. Tek gereksinim, işlevin dizinin her bir parçasından tek bir değer (azalma) üretmesidir. Örneğin, *rolling(...).quantile(q)* kullanarak numune nicelikleri hesaplayabilirken, numune üzerindeki belirli bir değerin yüzdelik sıralaması ile ilgilenebiliriz. *scipy.stats.percentileofscore* fonksiyonu sadece bunu yapar.



**BÖLÜM 12**

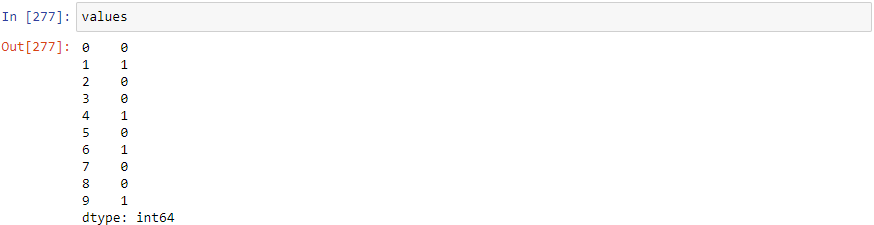
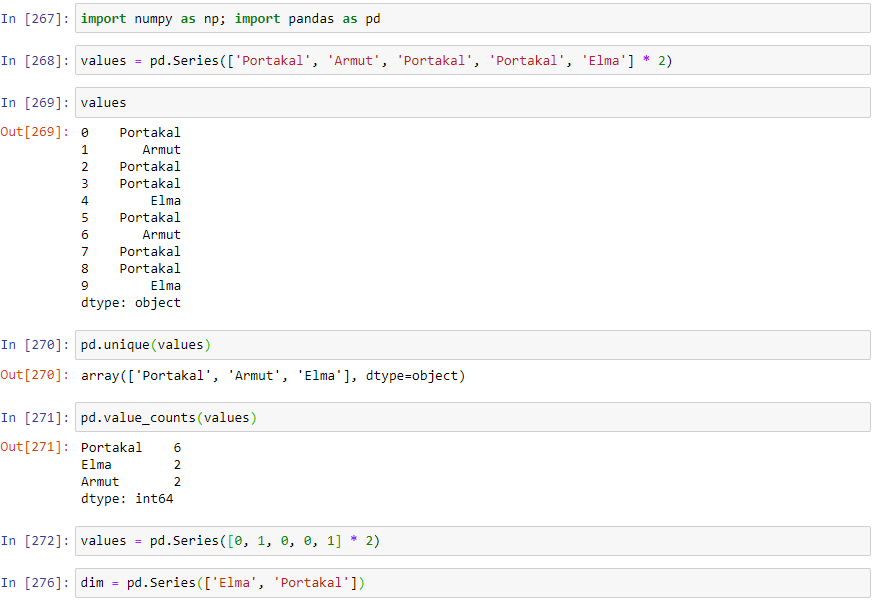
**GELİŞMİŞ PANDAS**

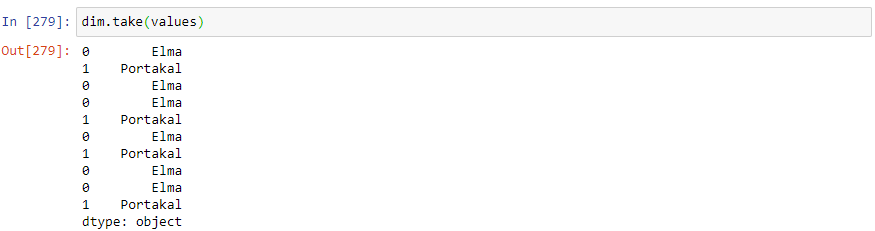
Önceki bölümler, NumPy, Pandas ve diğer kütüphanelerin iş akışlarını ve özelliklerini tartışan farklı veri türlerini tanıtmaya odaklanmıştır. Zamanla, Pandas uzman kullanıcılar için bir dizi özellik geliştirdi. Bu bölüm bir Pandas kullanıcısı olarak uzmanlığınızı derinleştirmenize yardımcı olmak için birkaç daha gelişmiş özellik alanına girmektedir.

**12.1. Kategorik Veriler**

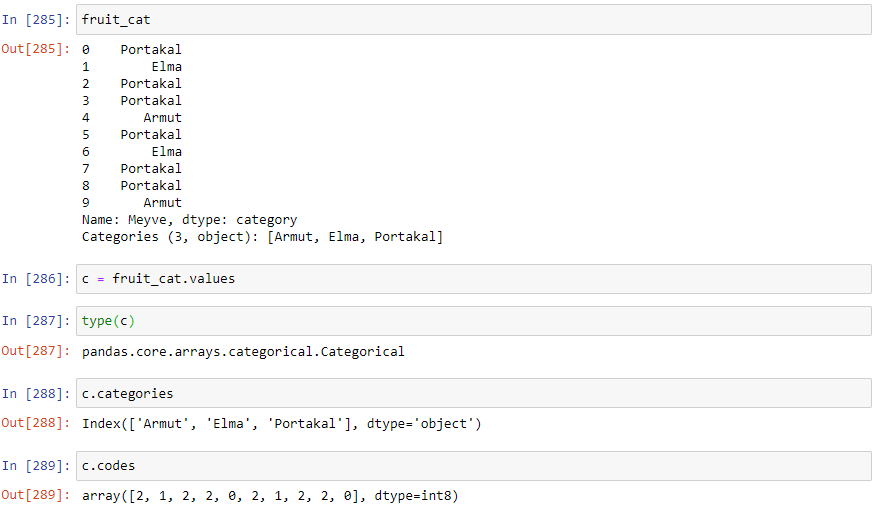
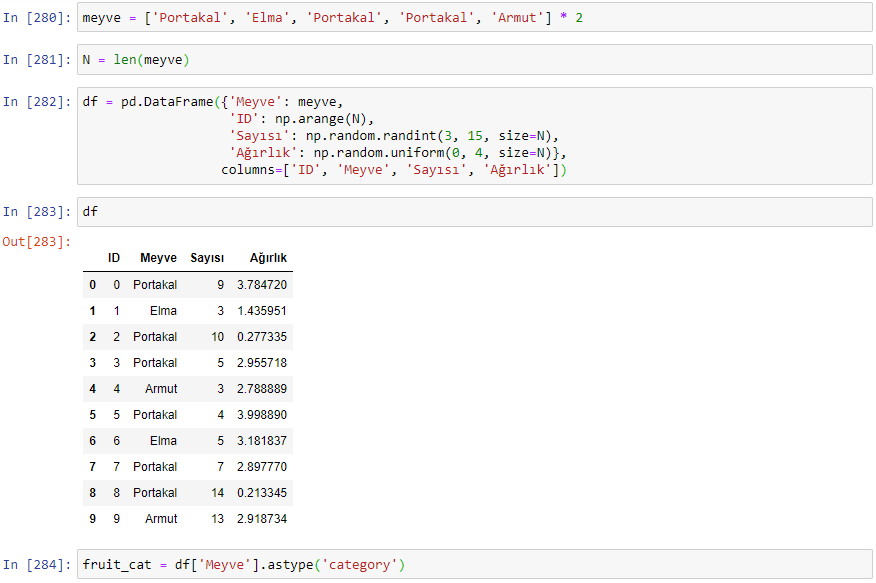
Bu bölüm Pandas’ın kategorik tipini tanıtmaktadır. Bazı Pandas operasyonlarında nasıl daha iyi performans ve hafıza kullanımı elde edebileceğinizi göstereceğim. Ayrıca kategorik verileri istatistiklerde ve makine öğrenim uygulamalarında kullanmak için bazı araçlar tanıtıyorum.

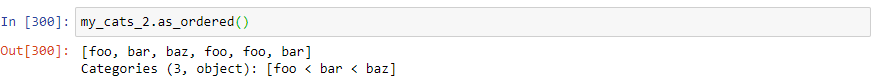
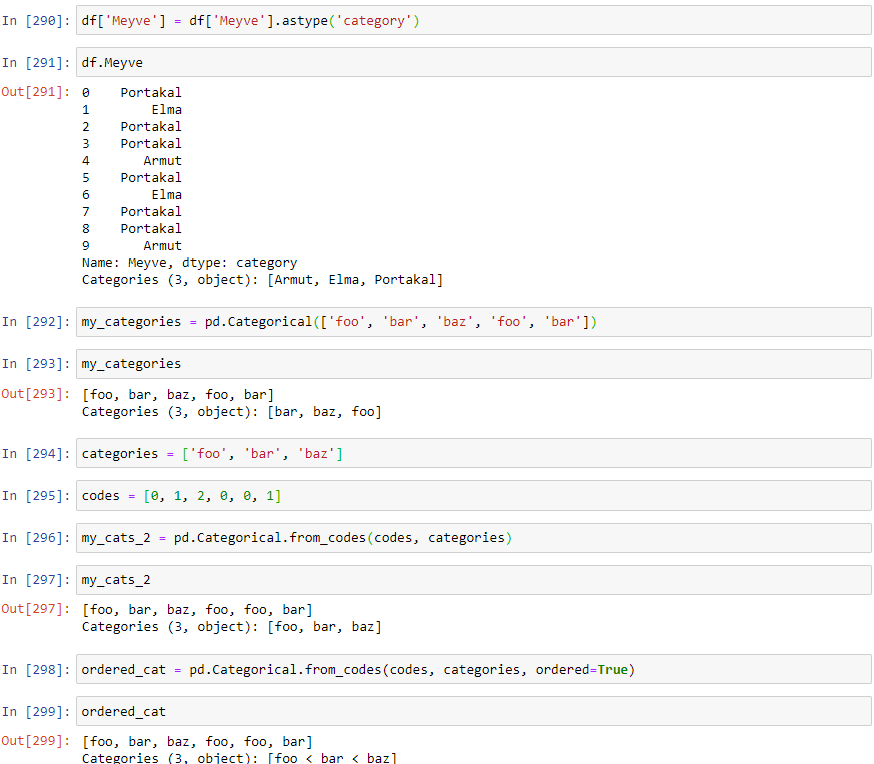
**Arka plan ve Motivasyon**



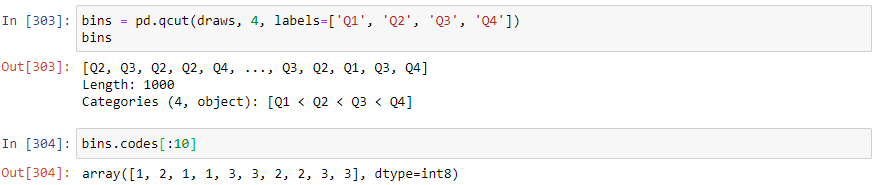
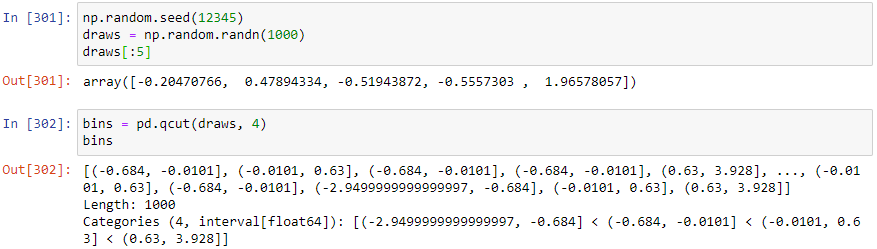


**Pandas’da Kategorik Tip**





**Kategorik Hesaplamalar**

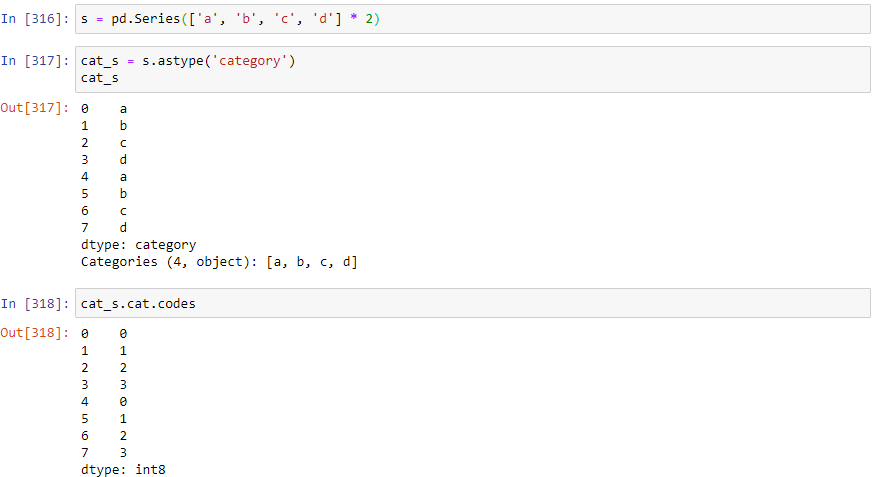


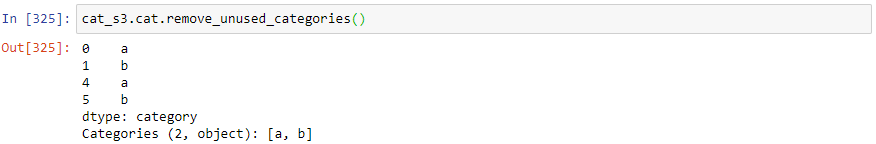
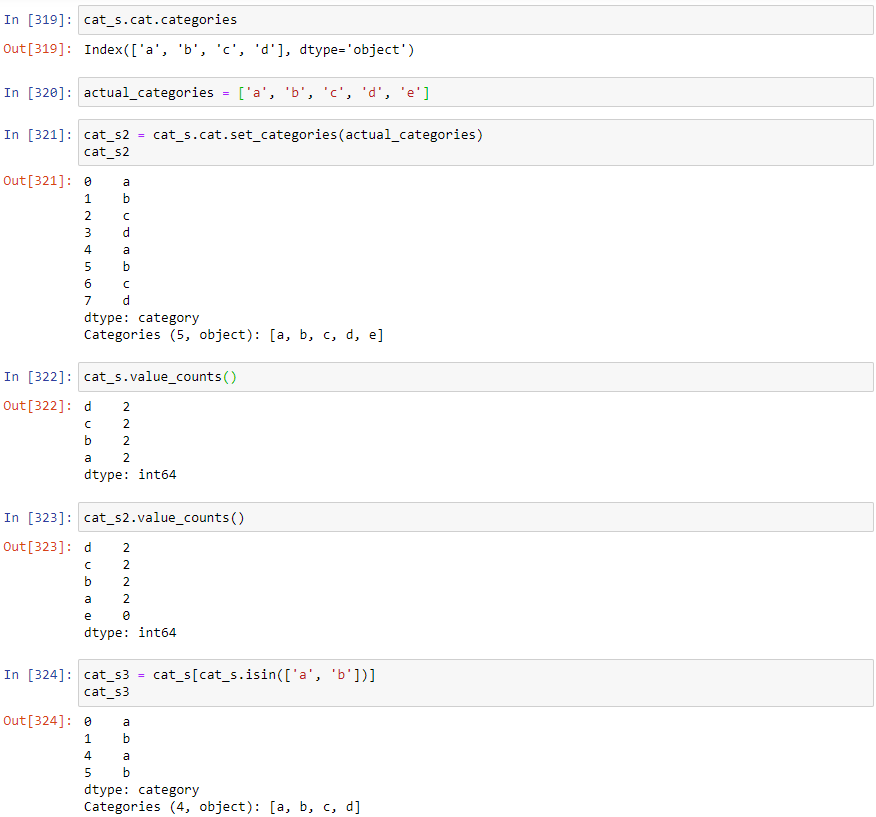


**Kategorilerle Daha İyi Performans**

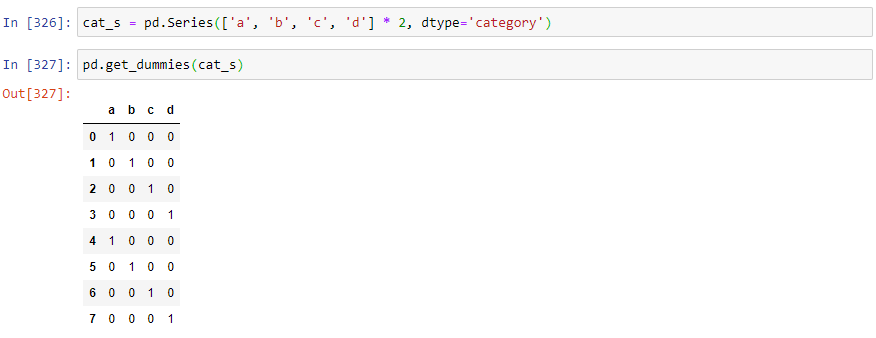


**Kategorik Yöntemler**



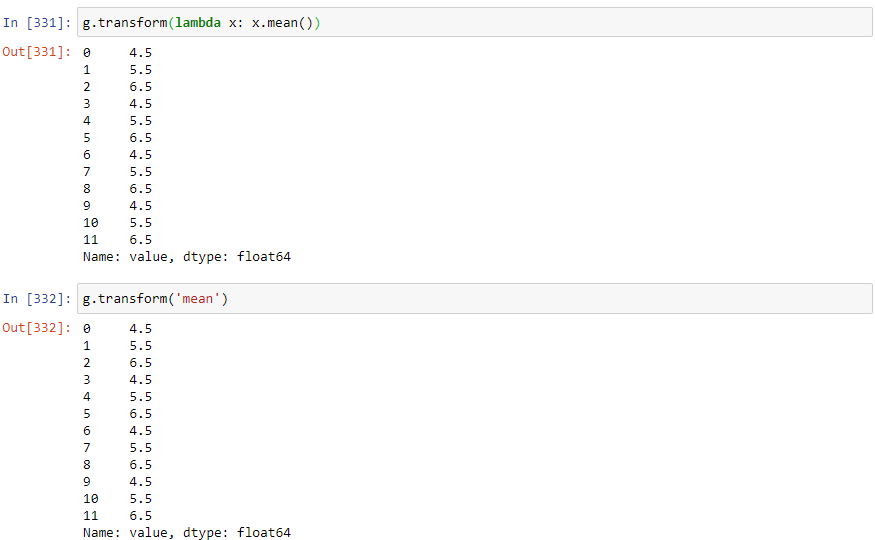
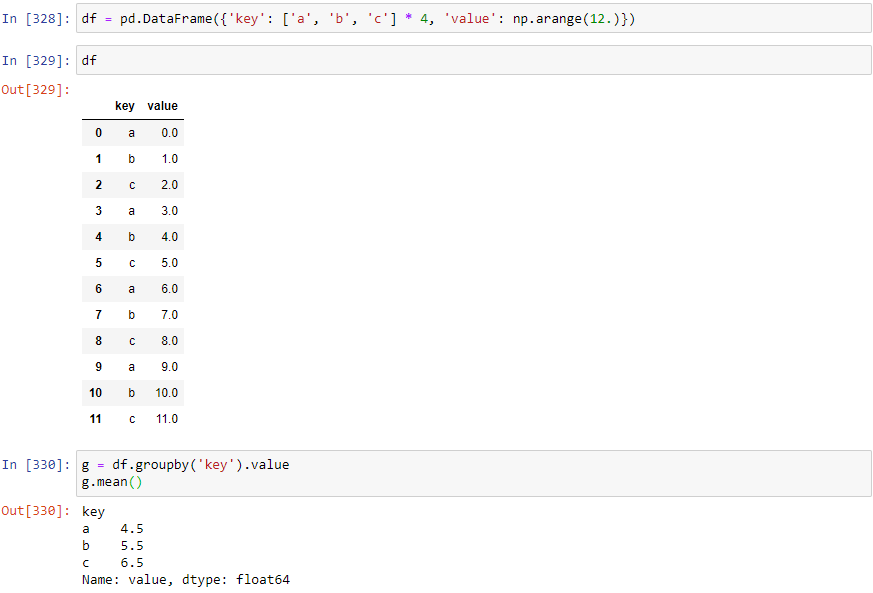


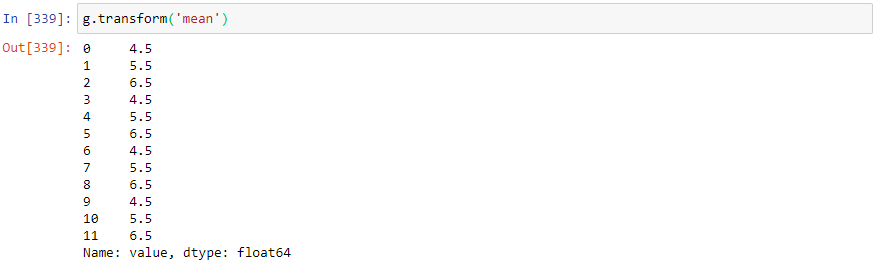
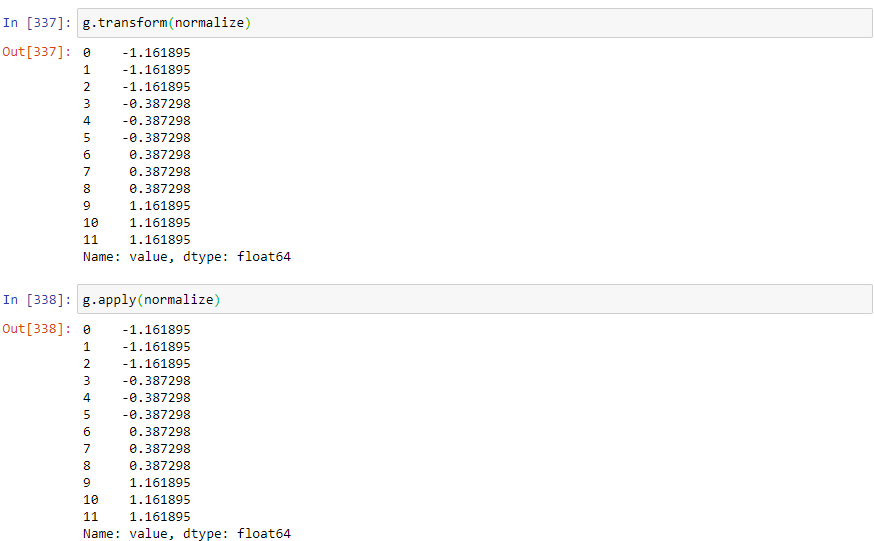
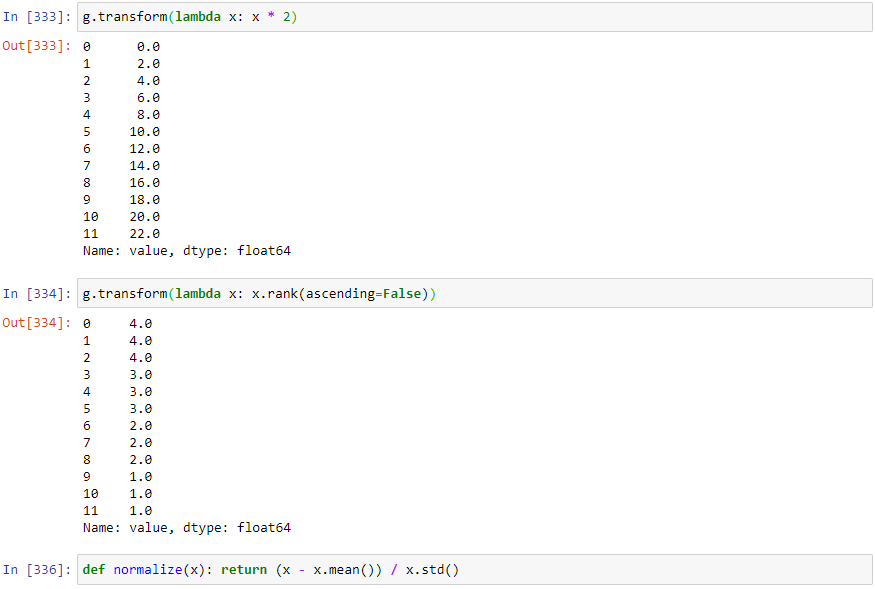
**Modelleme için Yapay Değişkenler Oluşturma**

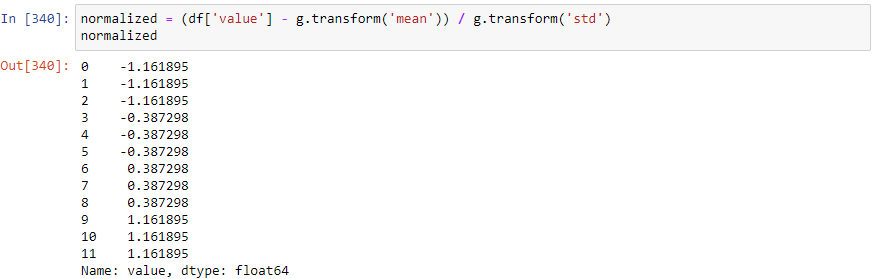


**12.2. Gelişmiş GroupBy Kullanımı**

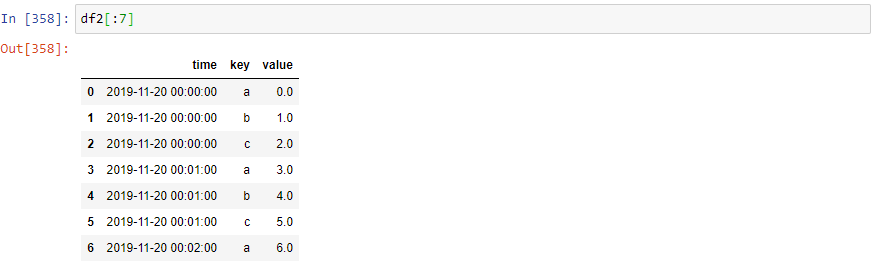
**Grup Dönüşümleri ve “Açılmamış” GroupBy**







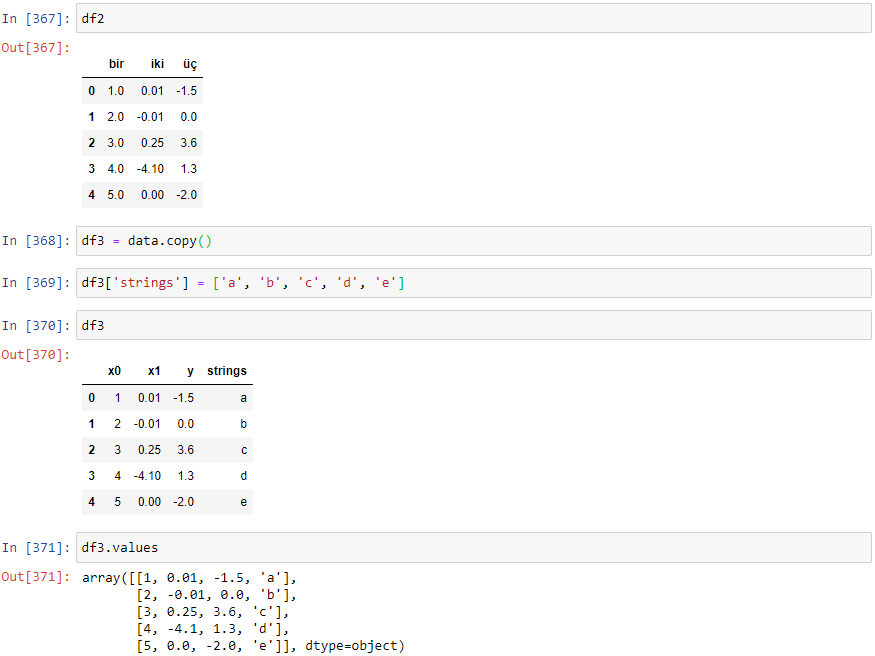
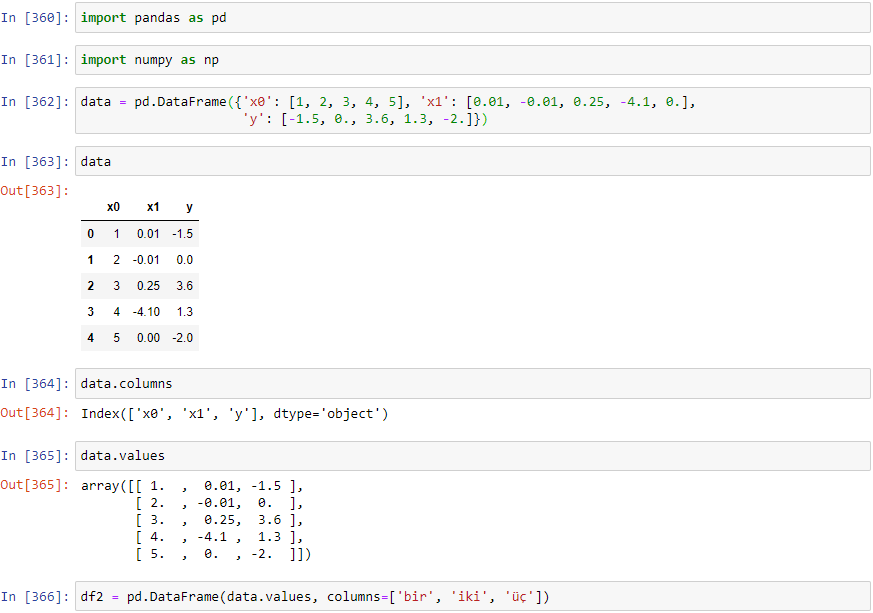
**Gruplandırılmış Zaman Yeniden Örnekleme**

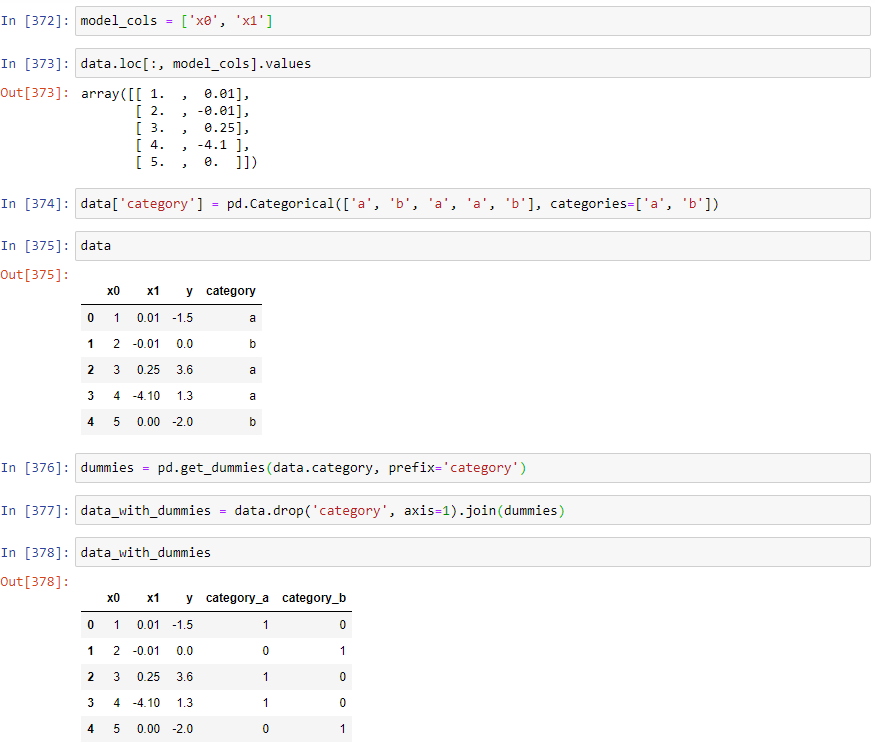


**BÖLÜM 13**

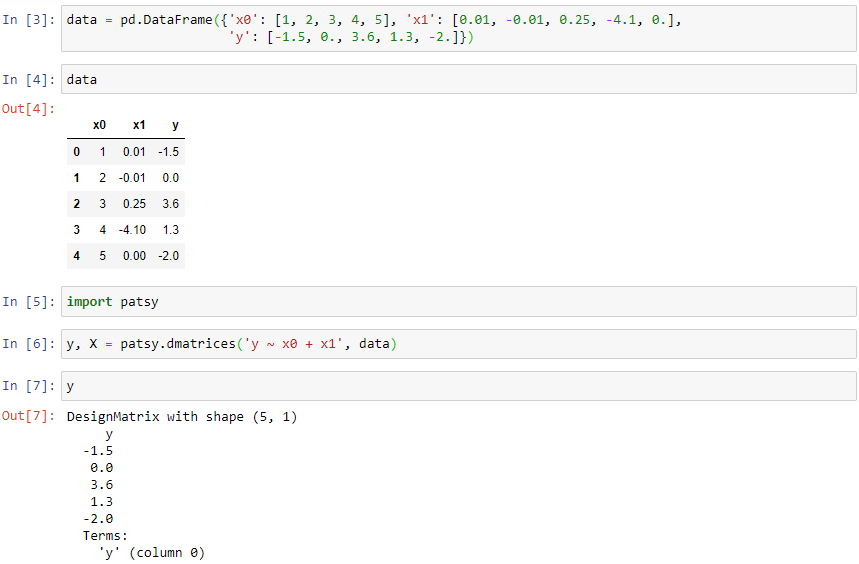
**Python'da Modelleme Kütüphanelerine Giriş**

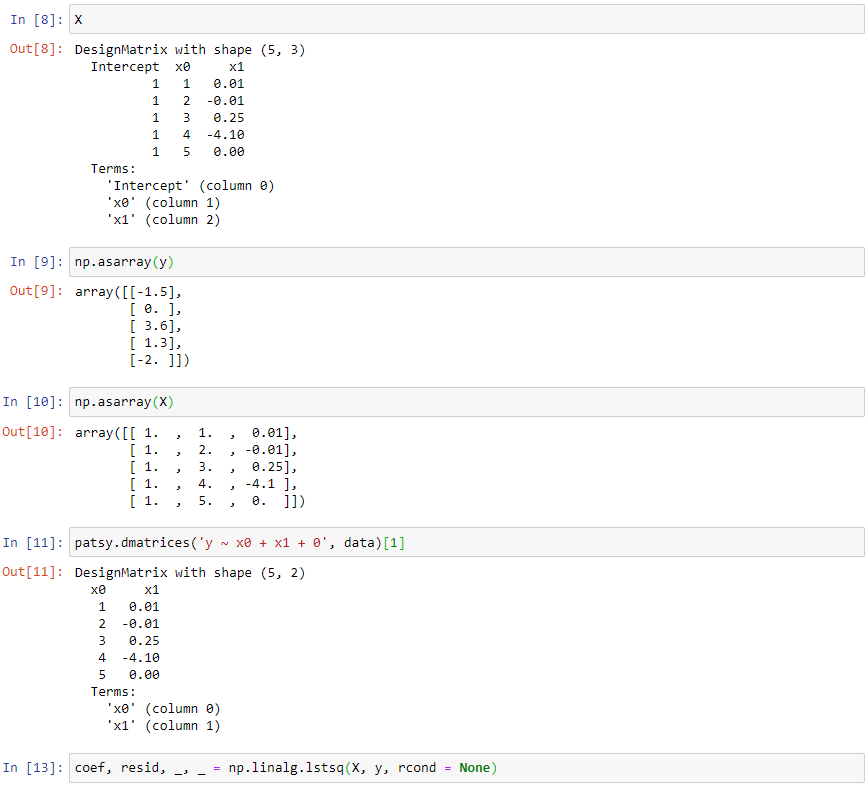
**13.1. Pandas ve Model Kodu Arasında Arabirim Oluşturma**



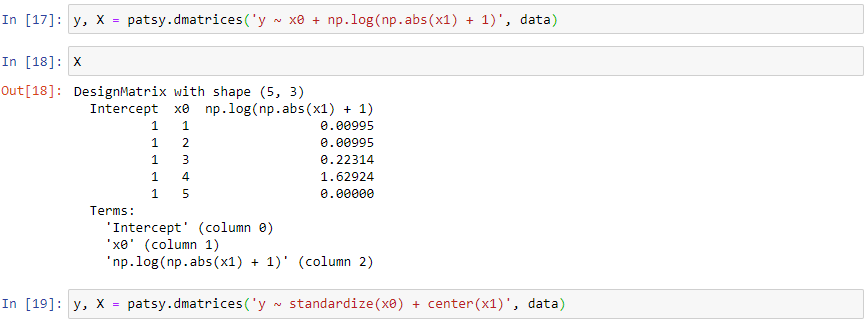


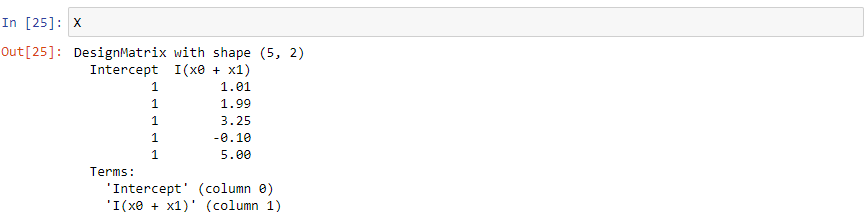
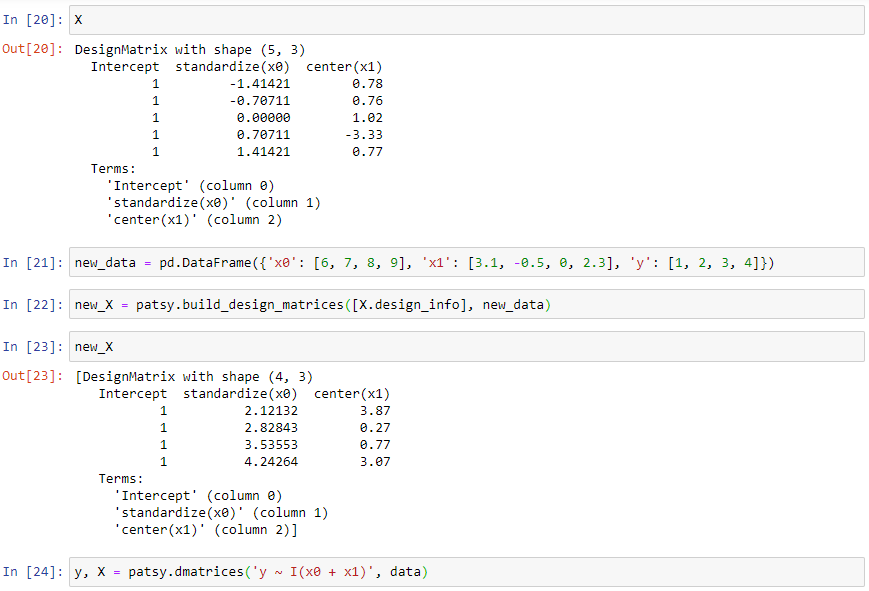
**13.2. Patsy ile Model Tanımları Oluşturma**



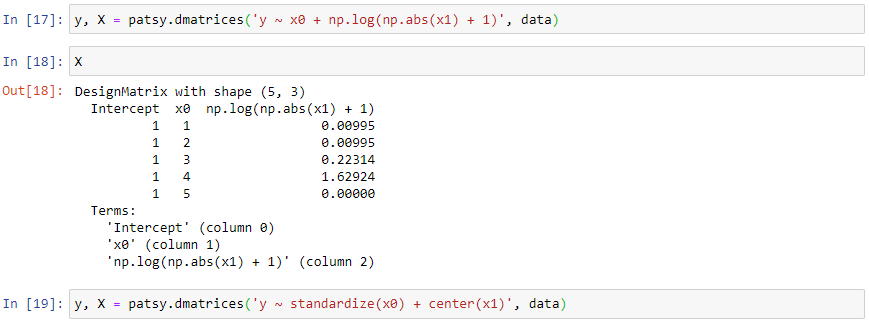


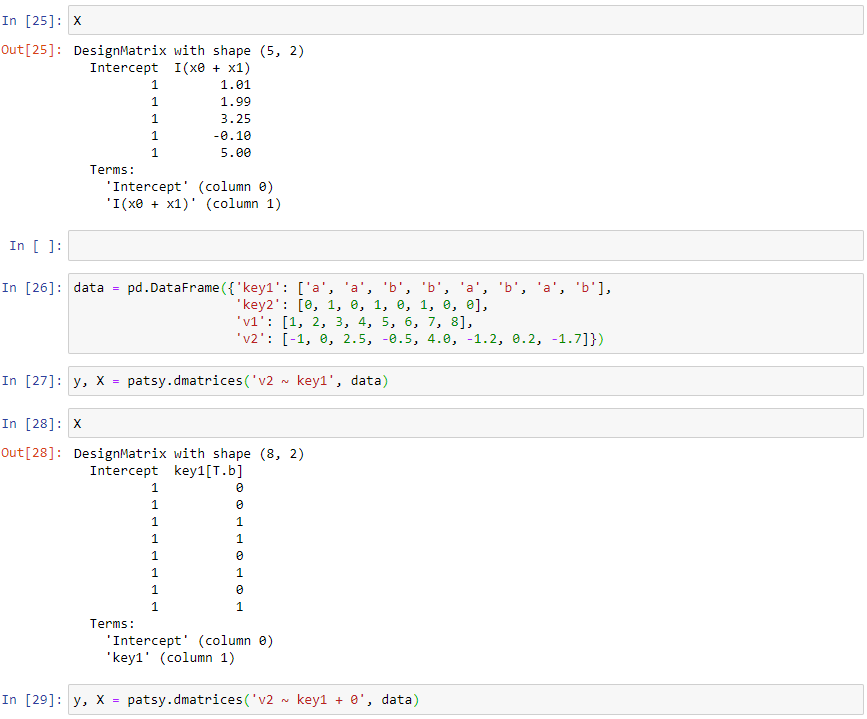
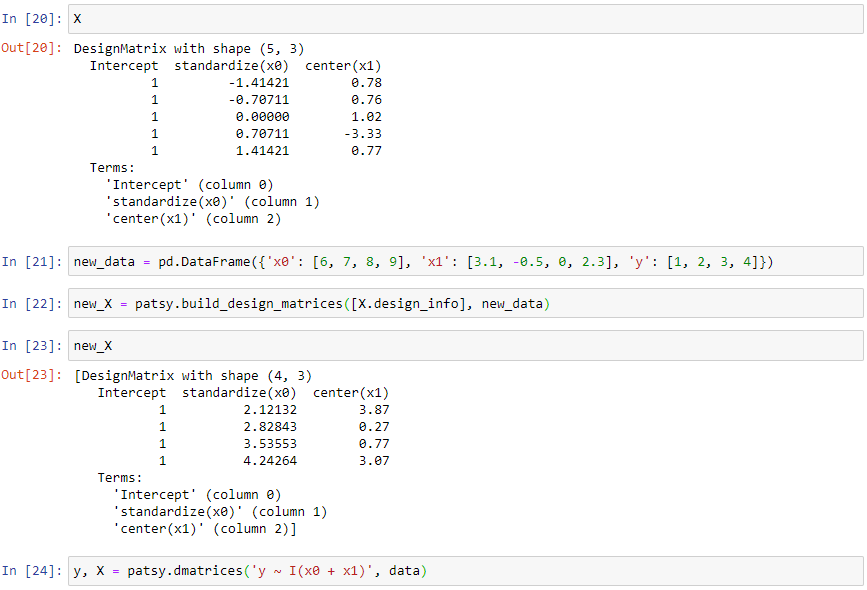
**Patsy Formüllerinde Veri Dönüşümleri**

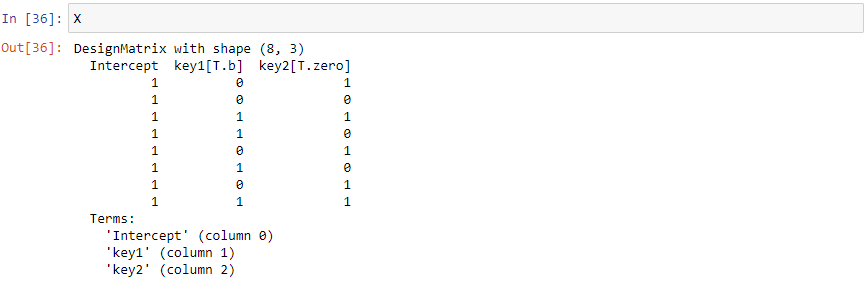
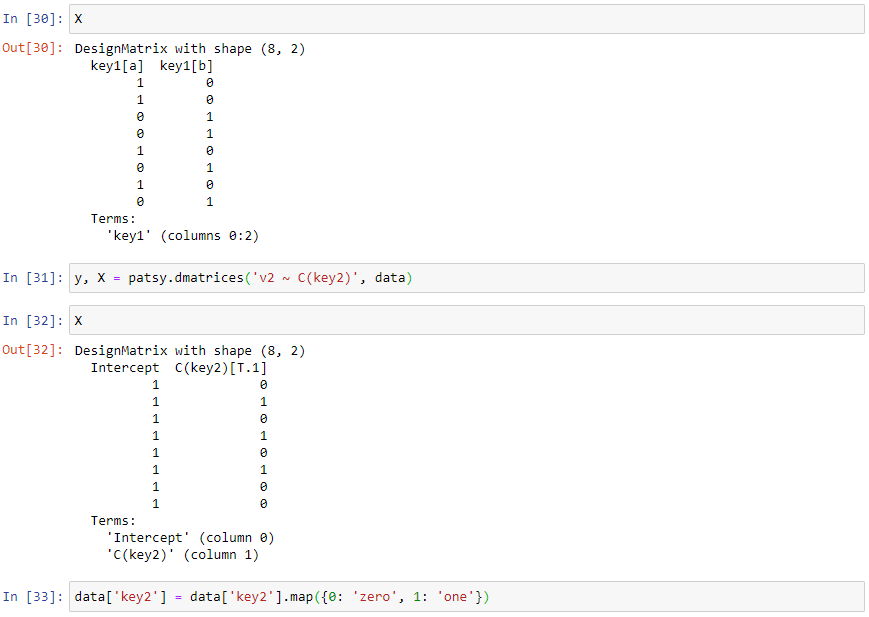


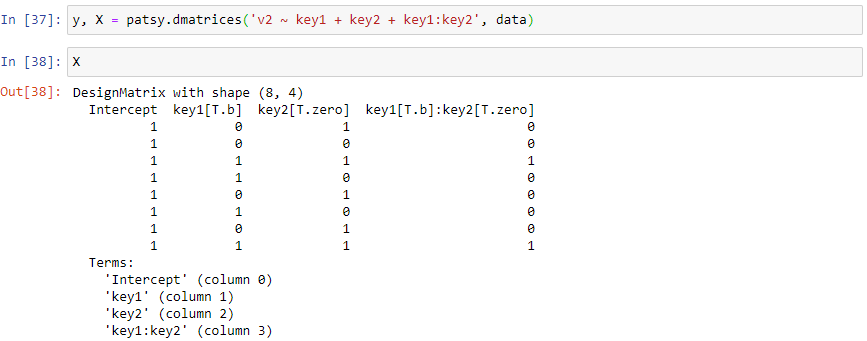


**Kategorik Veri ve Patsy**









**13.3. İstatistik Modellerine Giriş**

*statsmodels*, pek çok istatistiksel modelin yerleştirilmesi, istatistiksel testlerin yapılması, verilerin araştırılması ve görselleştirilmesi için kullanılan bir Python kütüphanesidir.

*statsmodels* de bulunan bazı model türleri şunlardır:

• Doğrusal modeller ve genelleştirilmiş doğrusal modeller

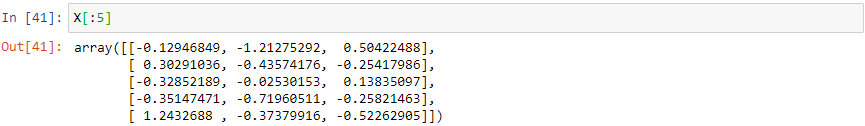
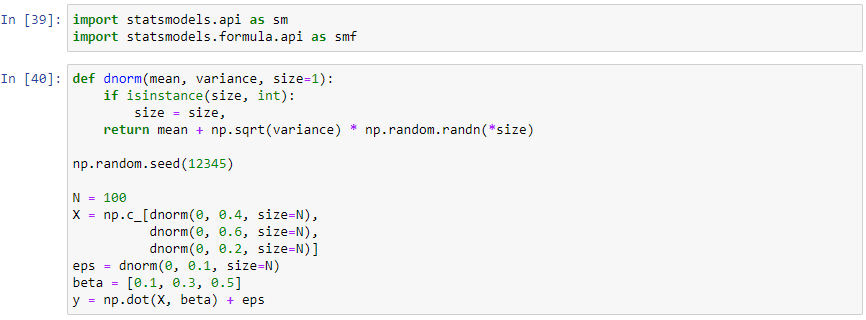
• Lineer karışık etki modelleri

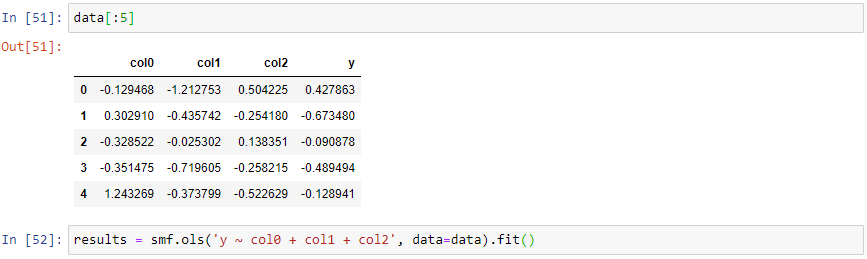
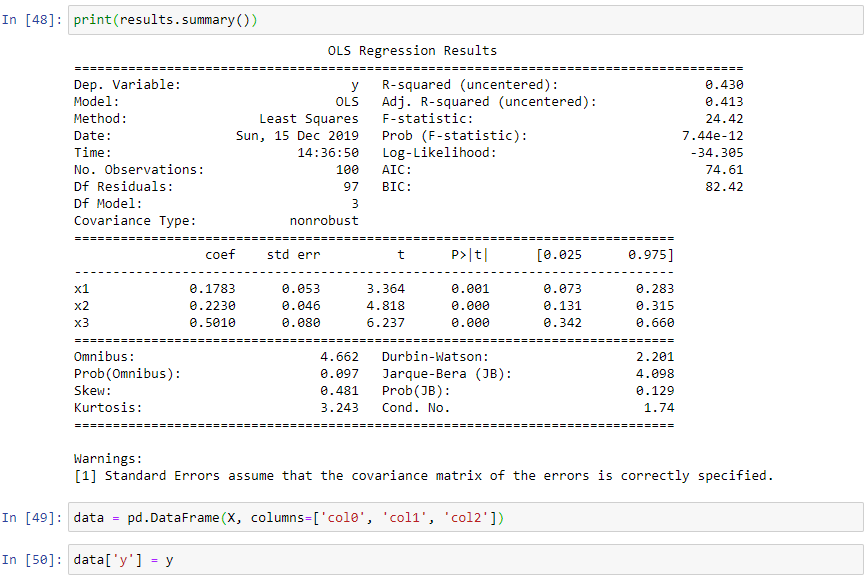
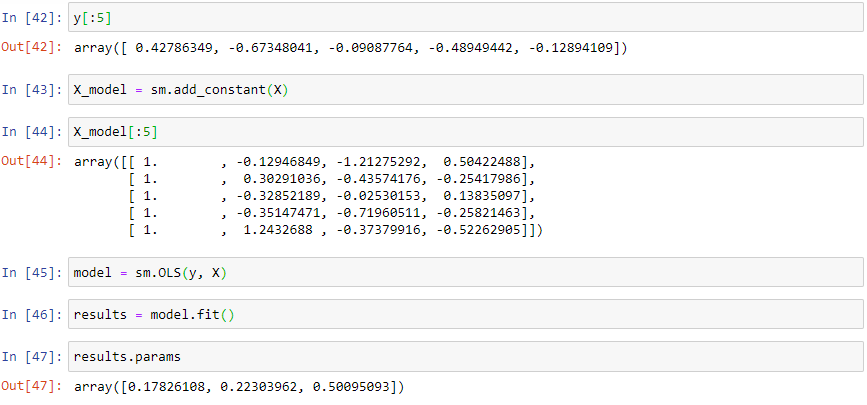
• Varyans analizi (ANOVA) yöntemleri

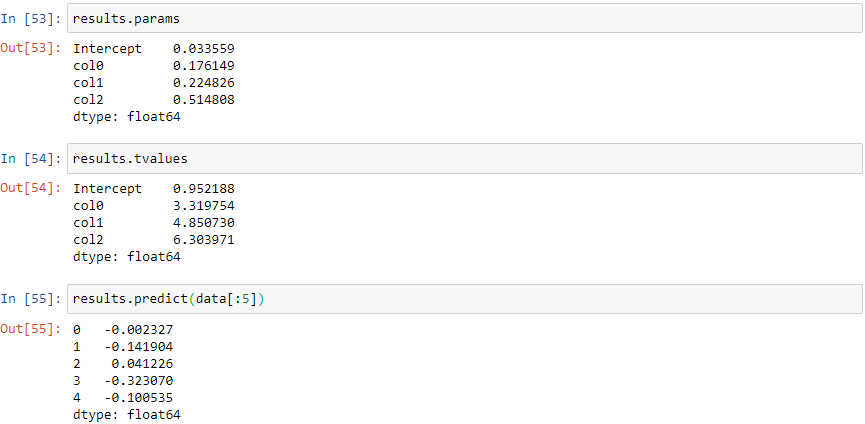
• Zaman serisi süreçleri ve durum uzayı modelleri

• Genelleştirilmiş momentler yöntemi

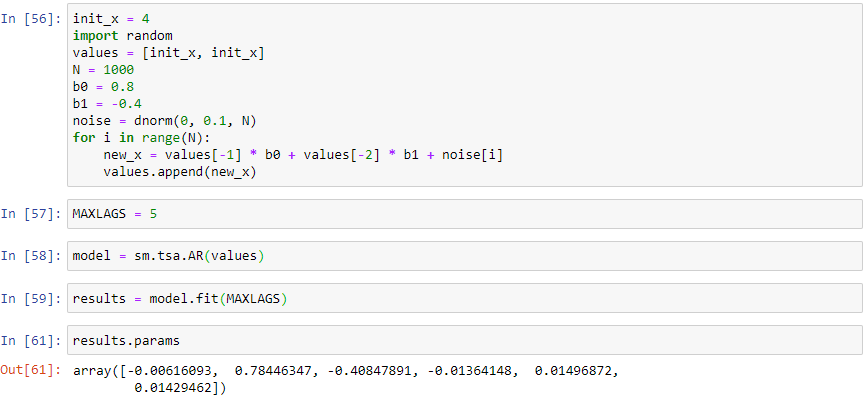
**Doğrusal Modellerin Tahmini**



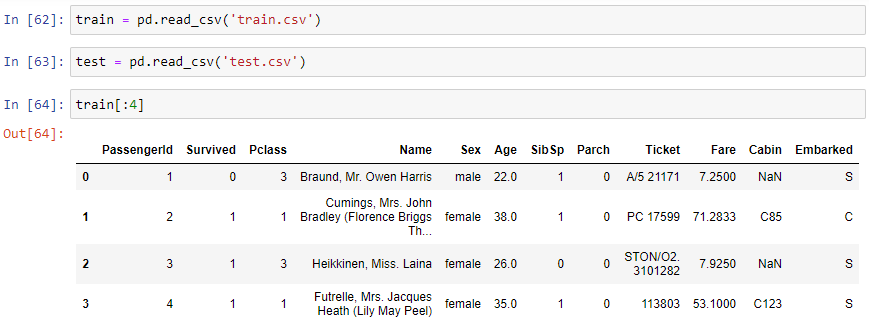




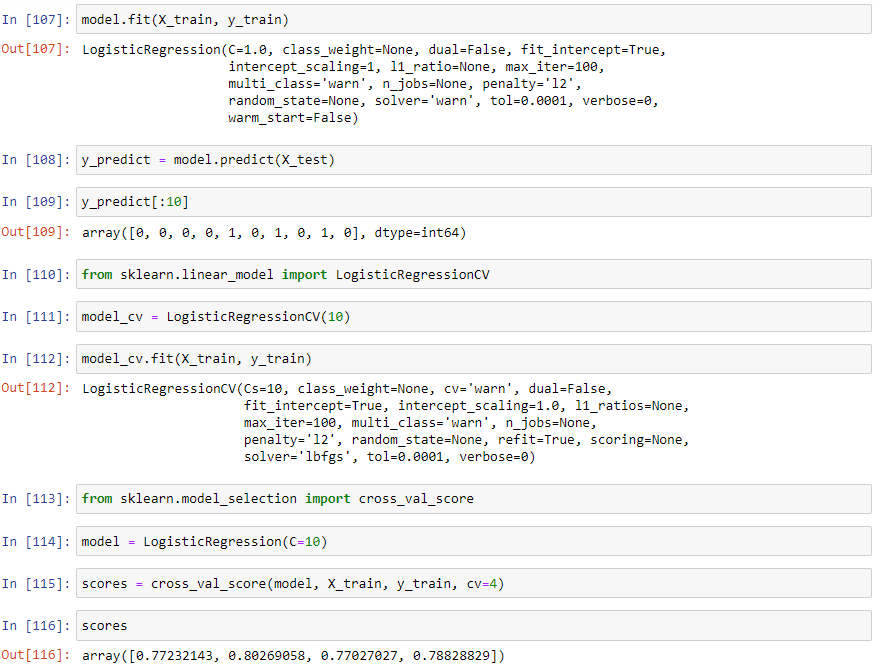
**Zaman Serisi Süreçlerini Tahmin Etme**



**13.4. Scikit-learn'e Giriş**



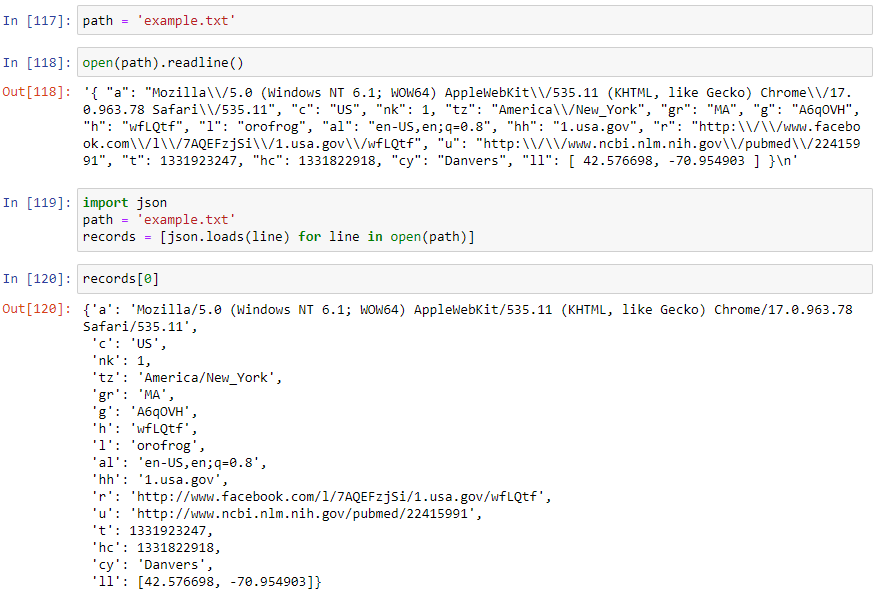




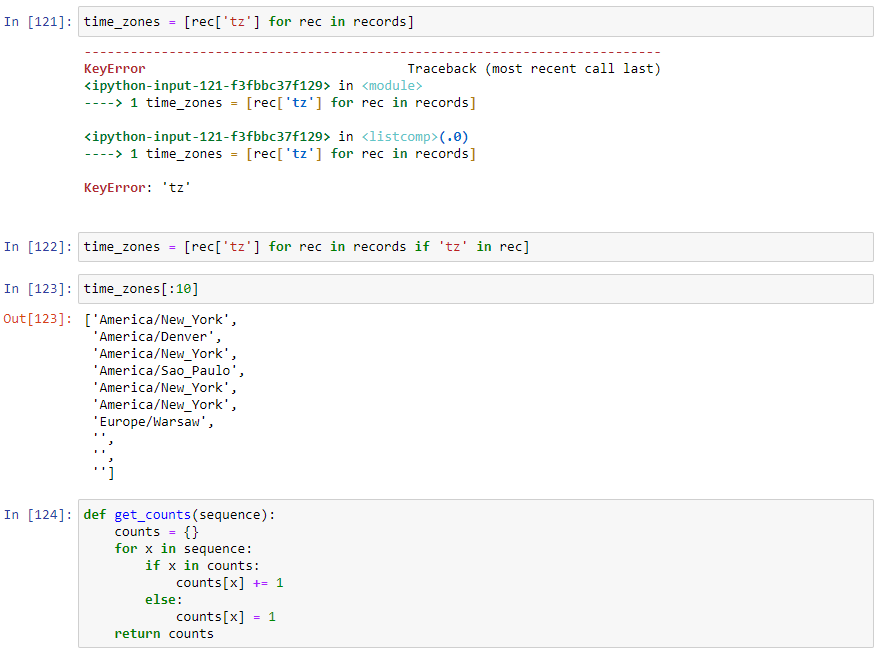
**BÖLÜM 14**

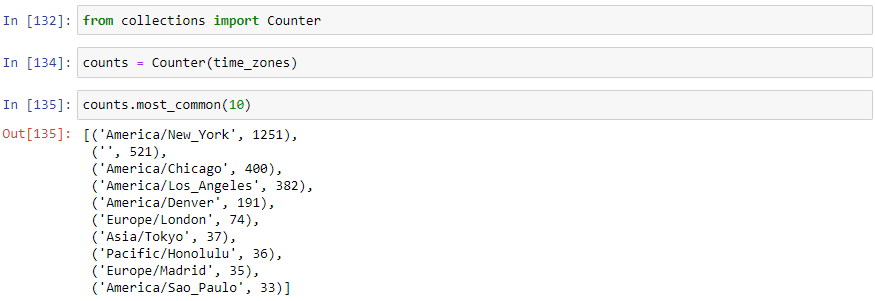
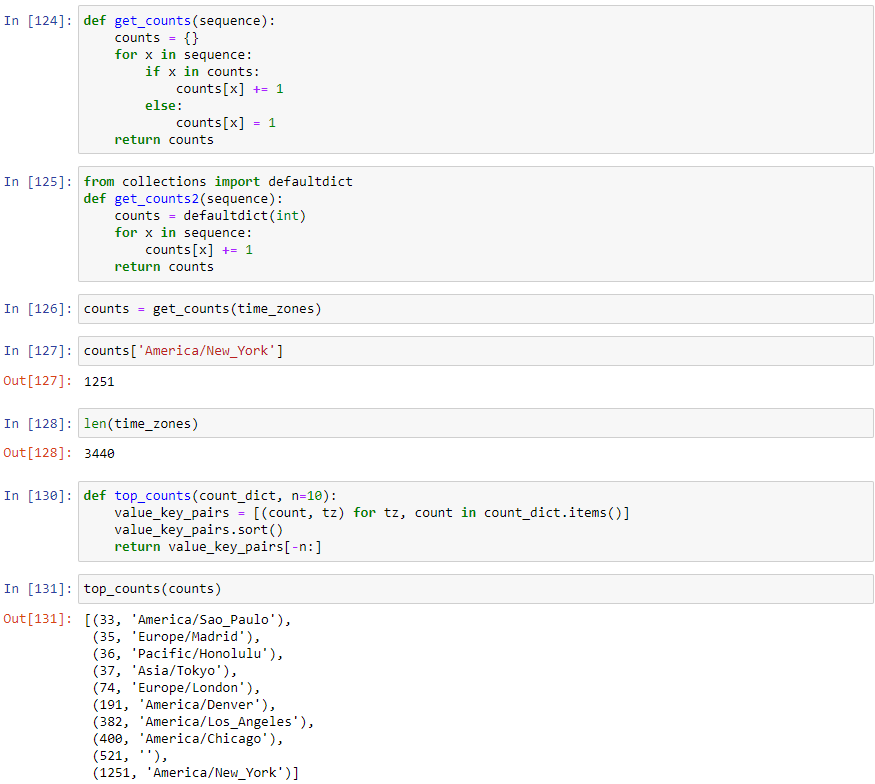
**Veri Analizi Örnekleri**

**14.1. USA.gov Data from Bitly**



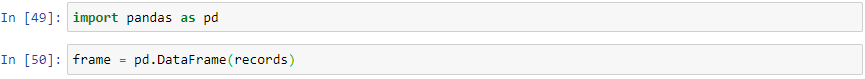
**Saf Python'da Zaman Dilimlerini Sayma**

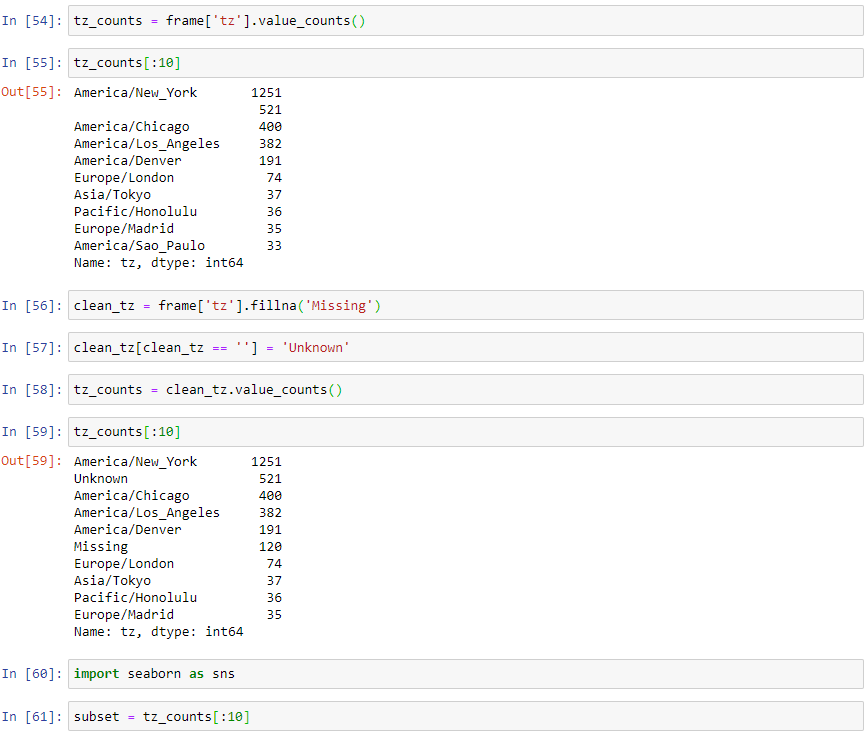
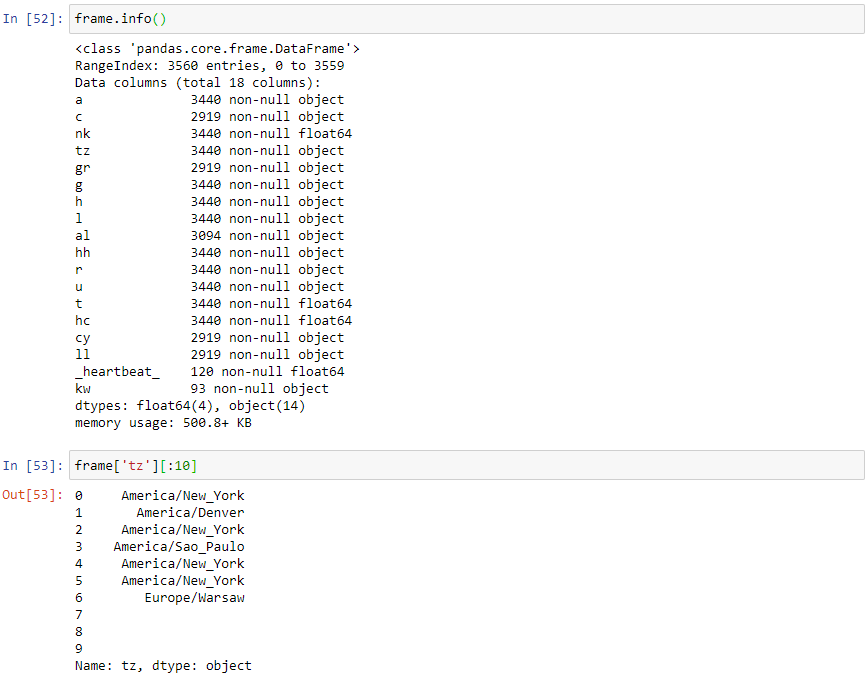


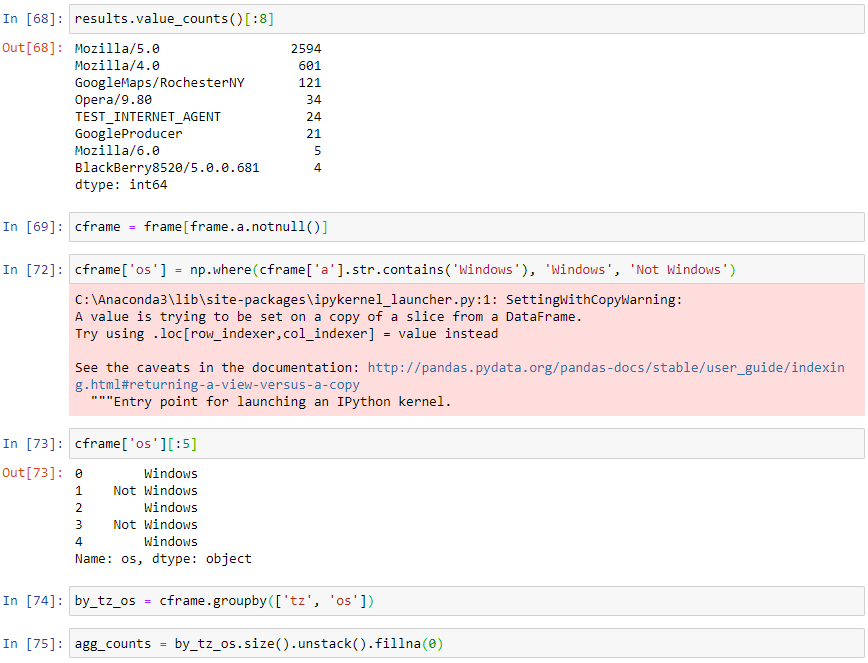
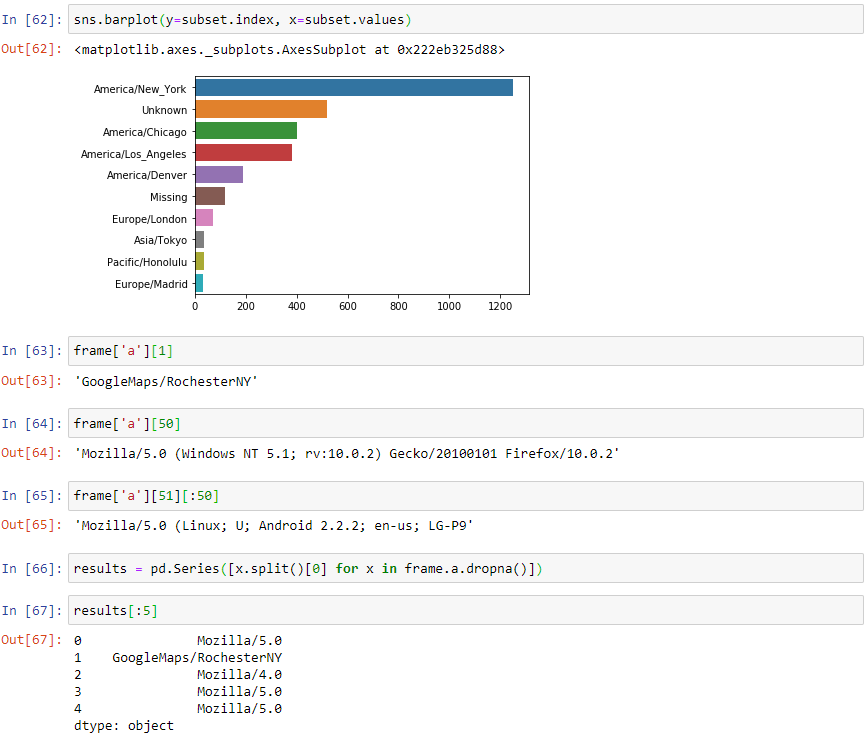


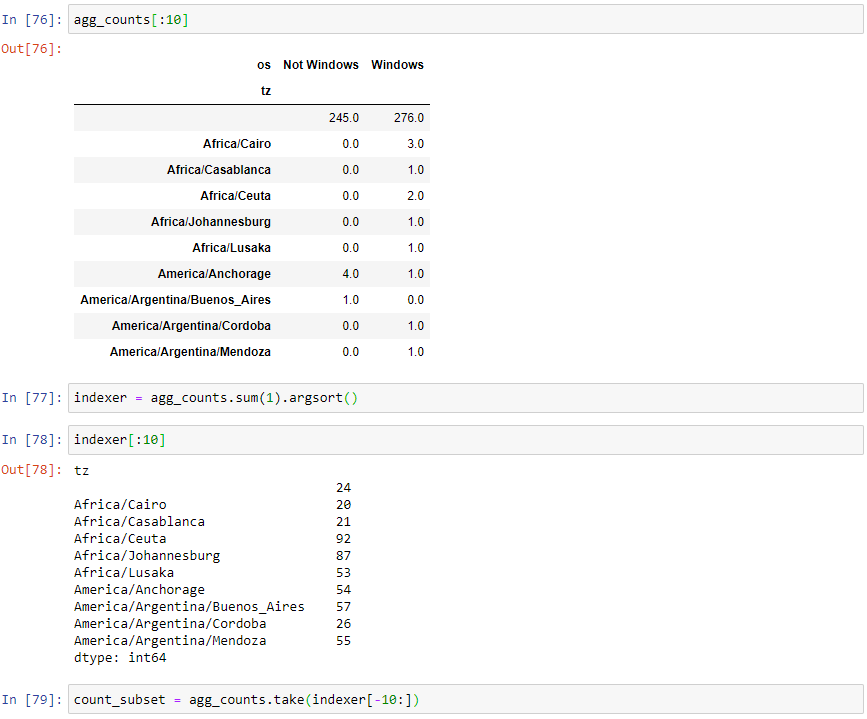
**Pandas ile Saat Dilimlerini Sayma**

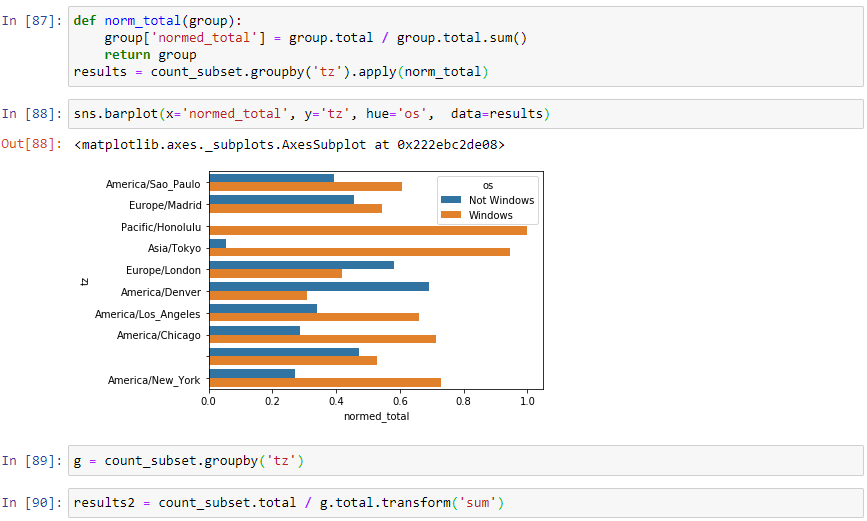
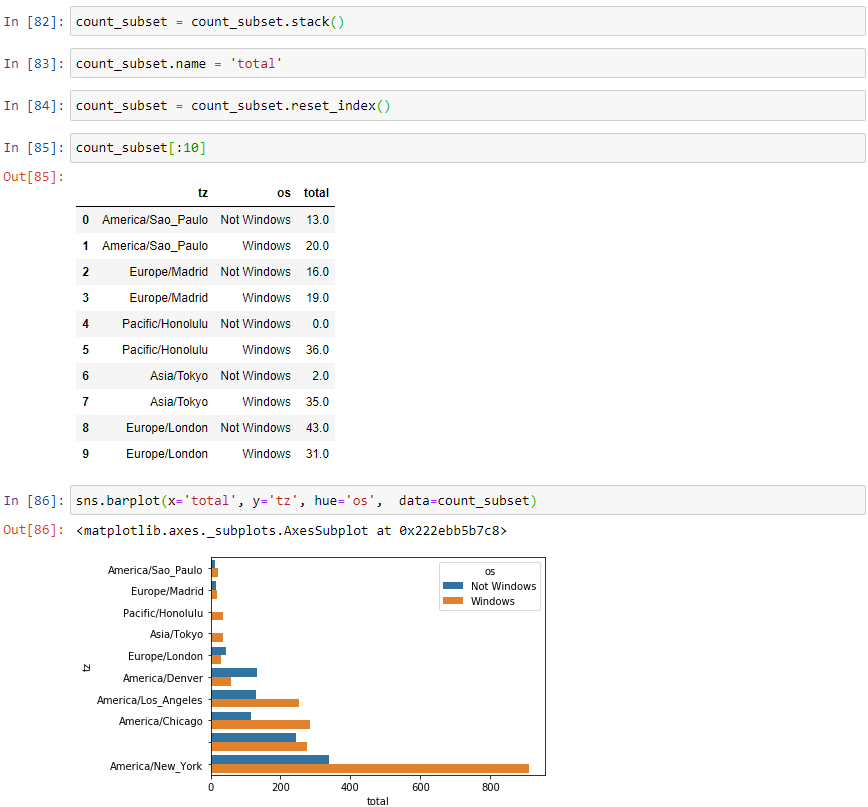
Orijinal kayıt kümesinden bir DataFrame oluşturmak, kayıt listesini pandasa aktarmak kadar kolaydır.



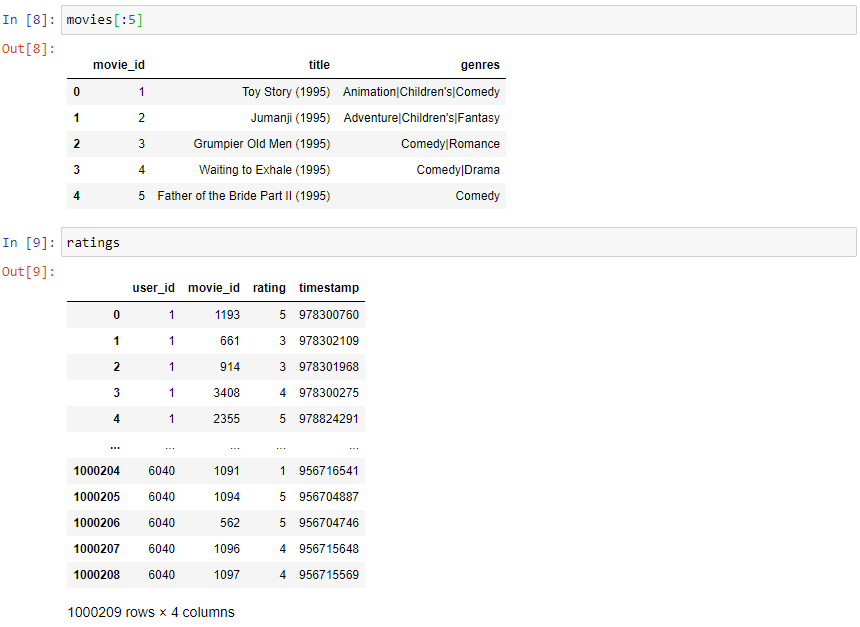
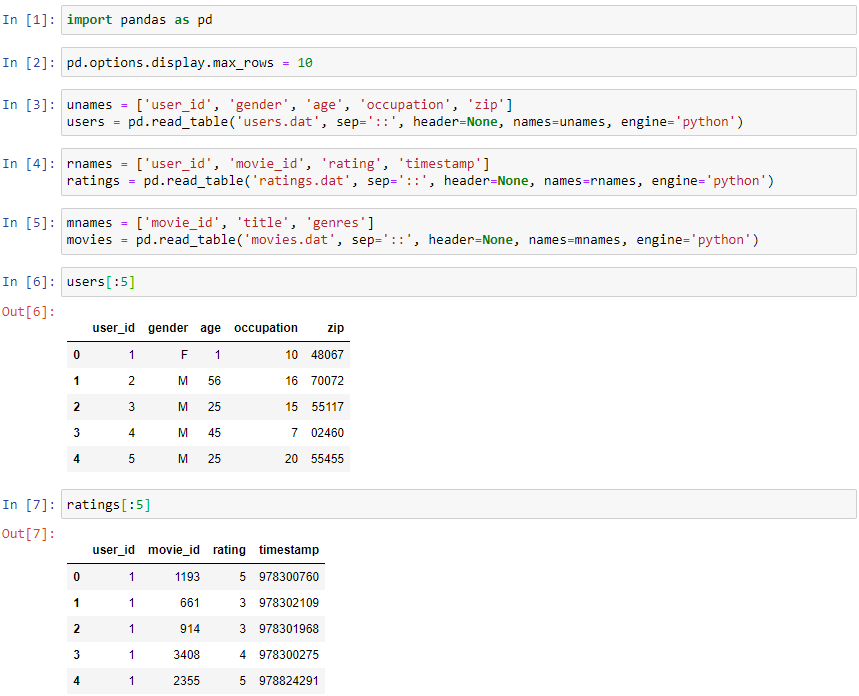


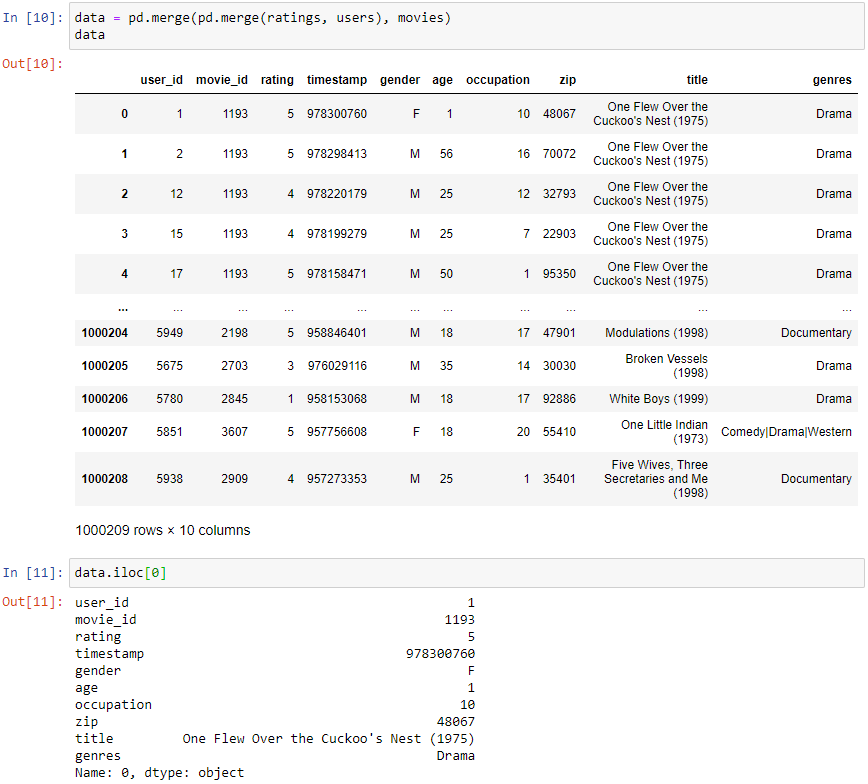


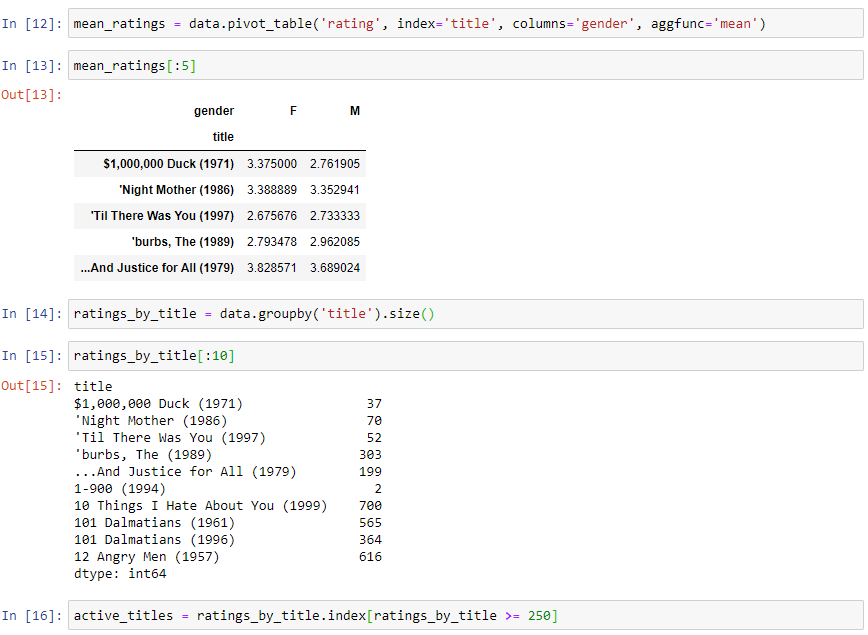


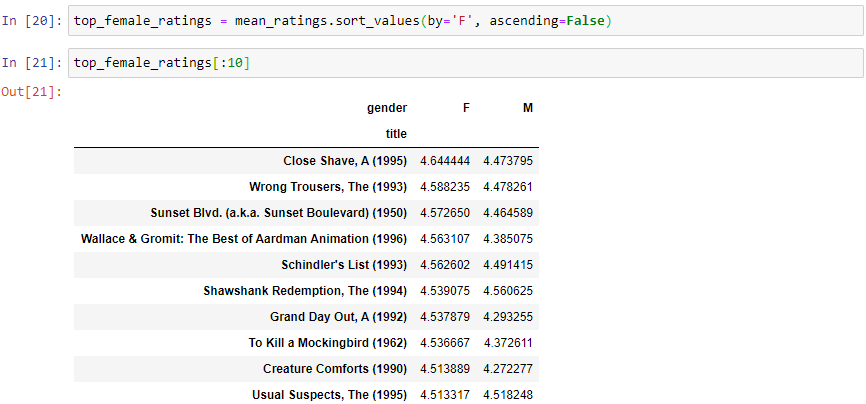
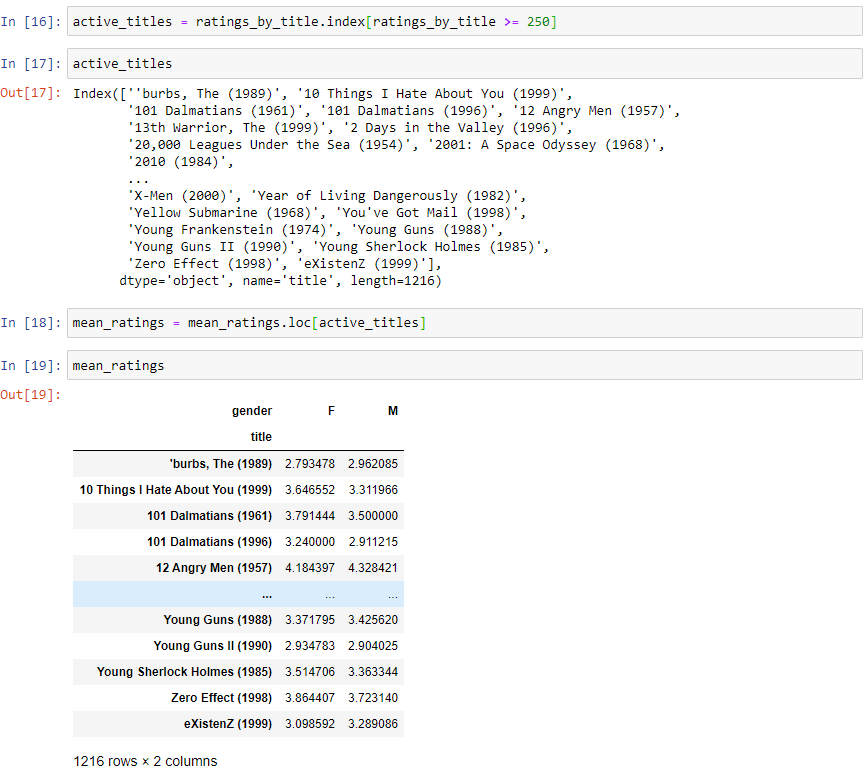


**14.2. MovieLens 1M Veri Kümesi**

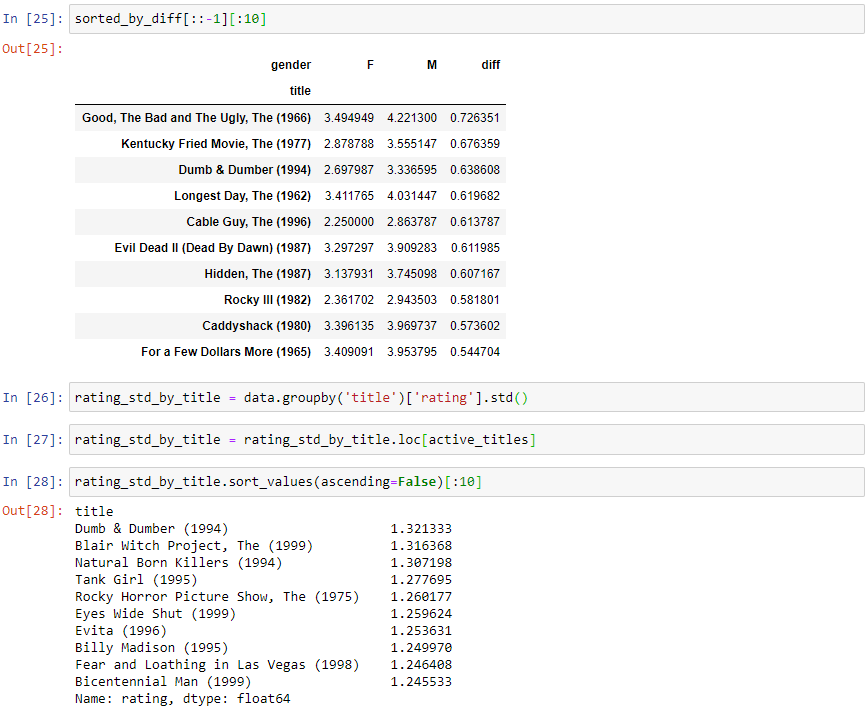
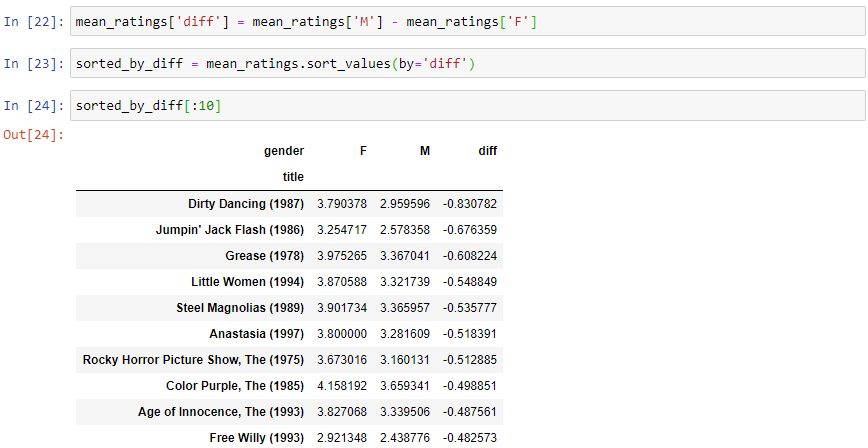




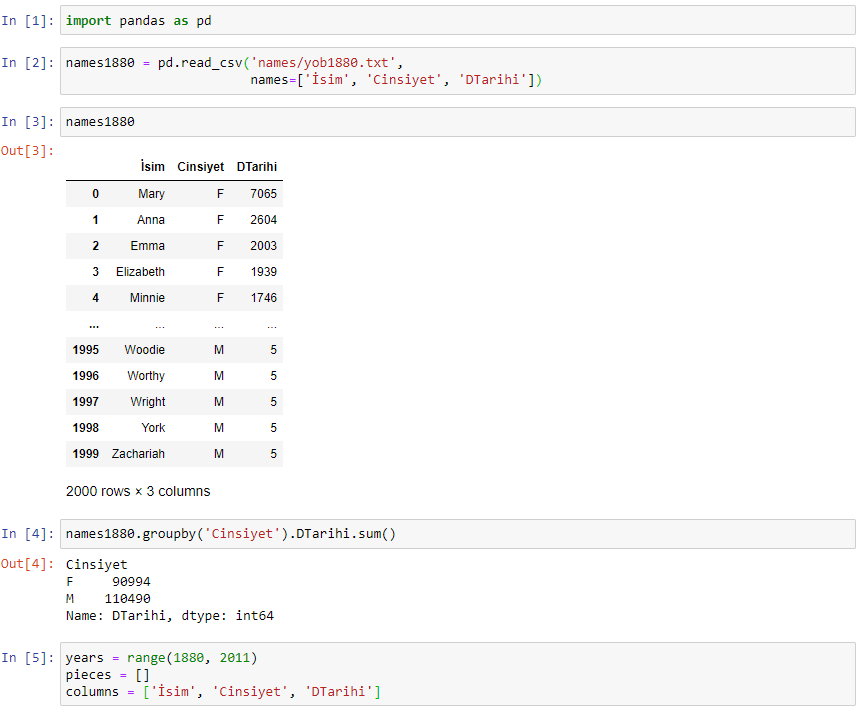


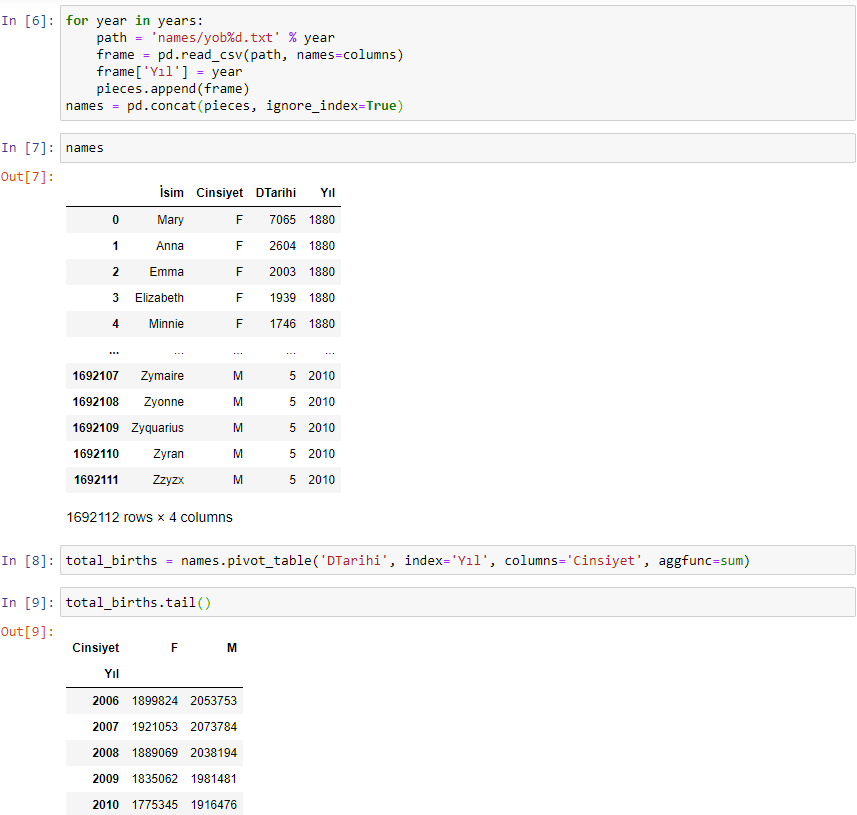


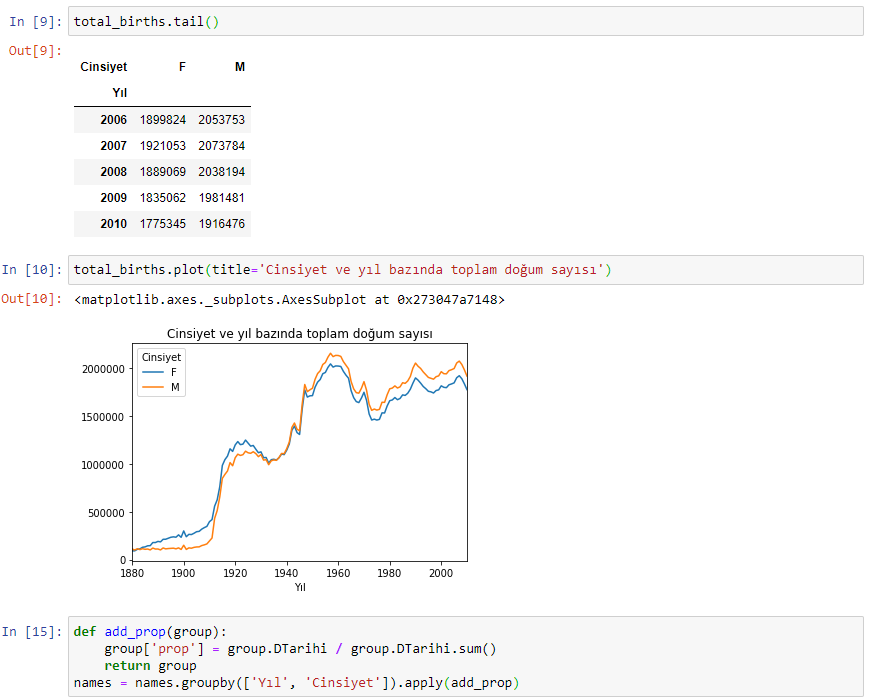
**Derecelendirme Anlaşmazlıklarını Ölçme**

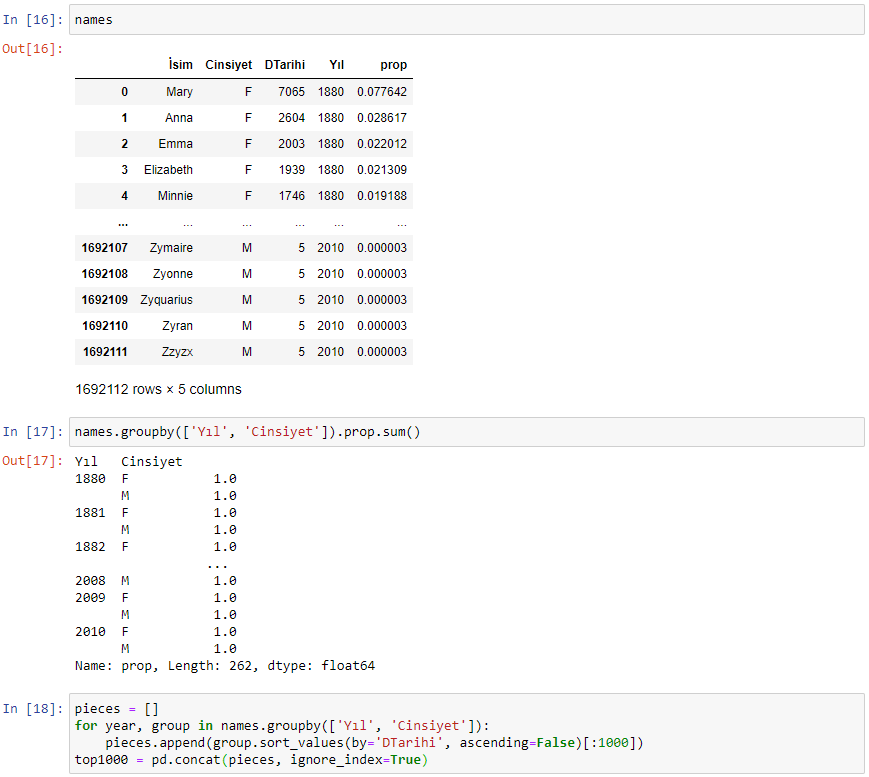


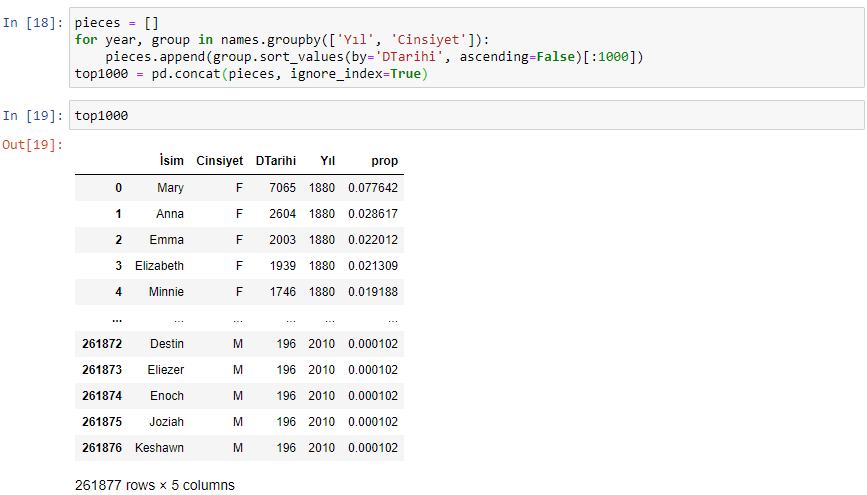
**14.3. ABD Bebek İsimleri 1880–2010**



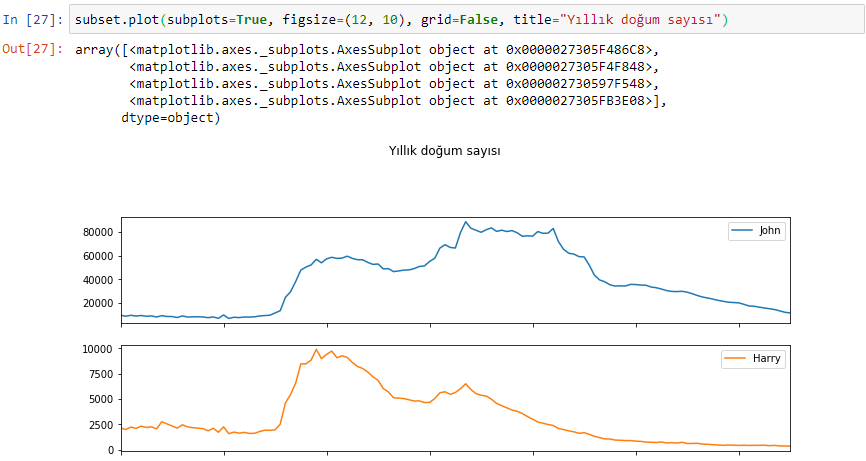
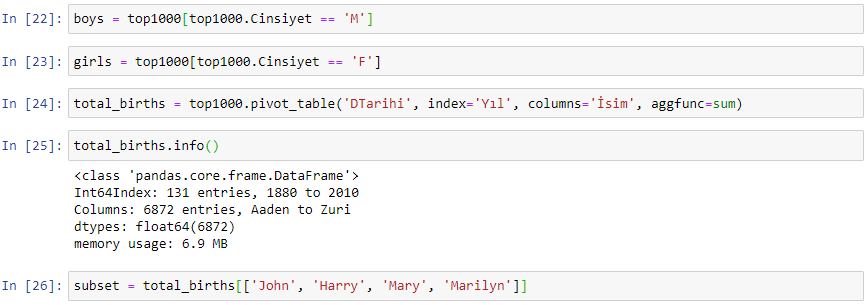






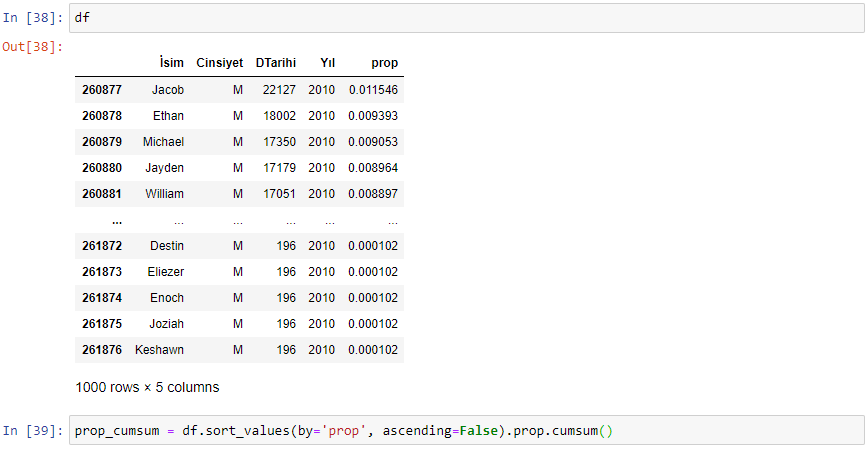
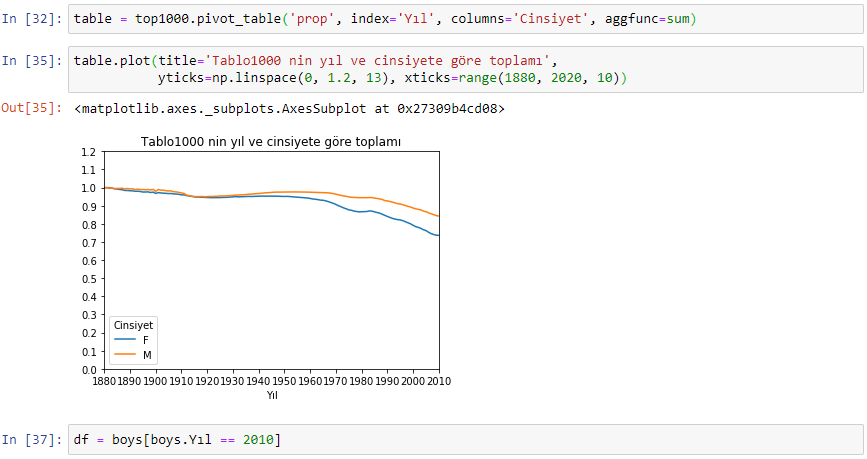


**Adlandırma Trendlerini Analiz Etme**

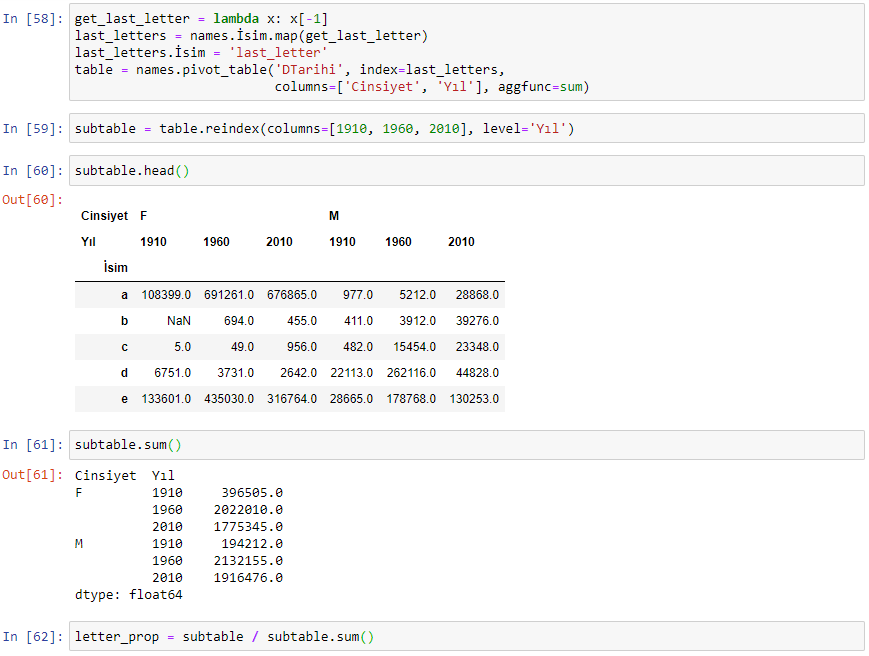


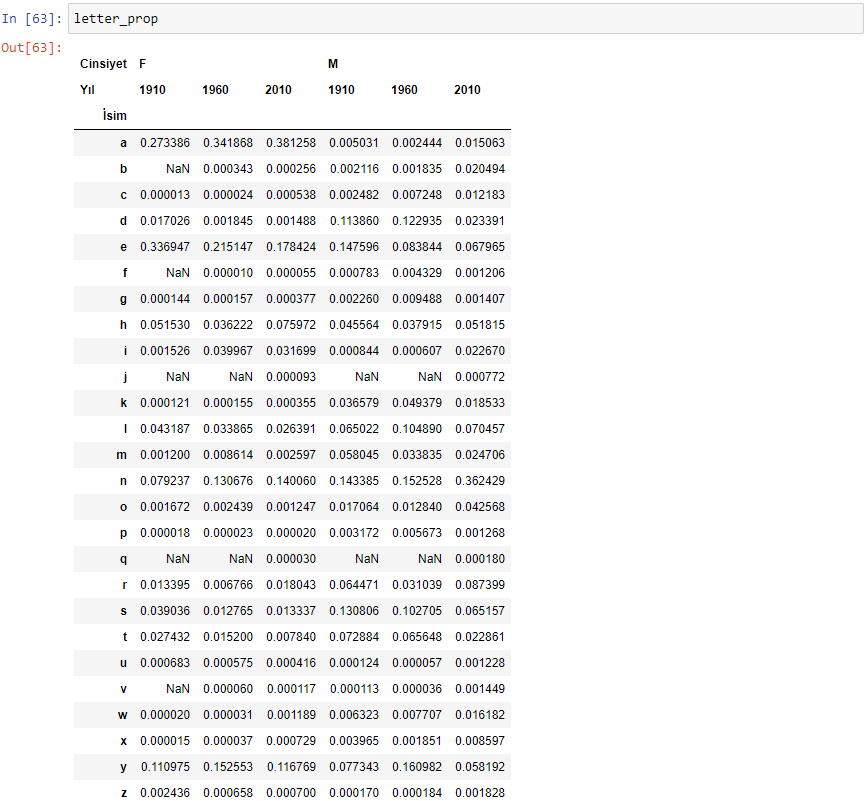


**Adlandırma Çeşitliliğindeki Artışın Ölçülmesi**

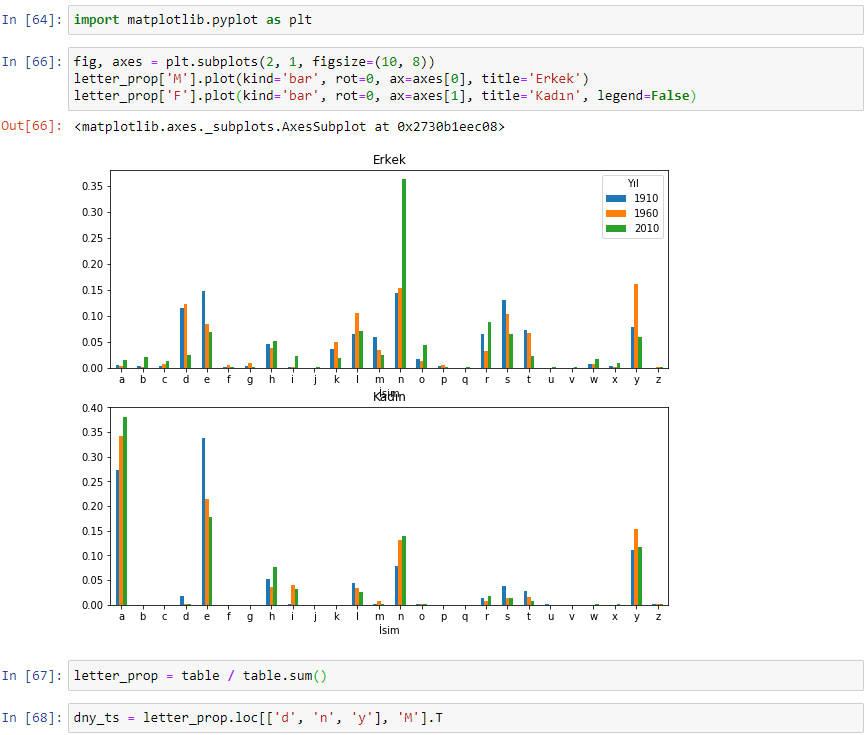


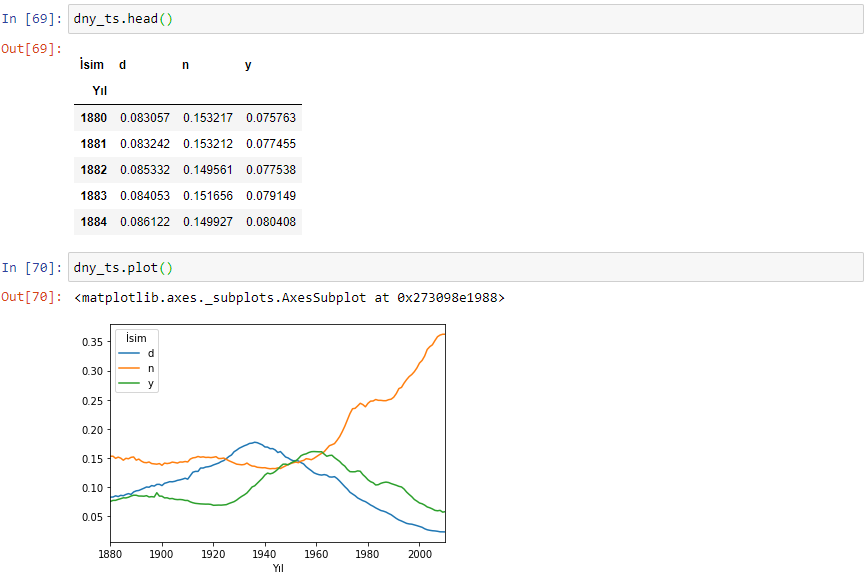


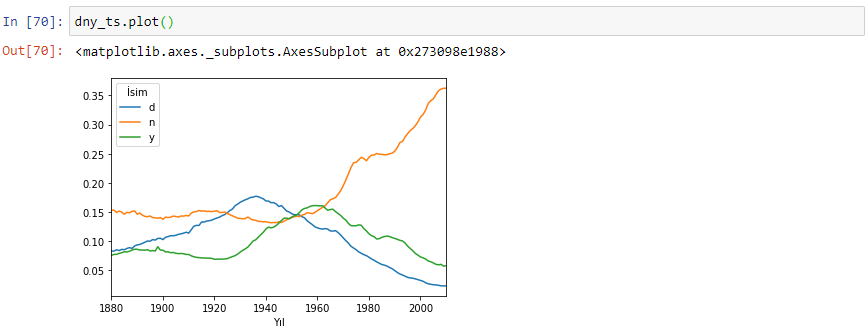




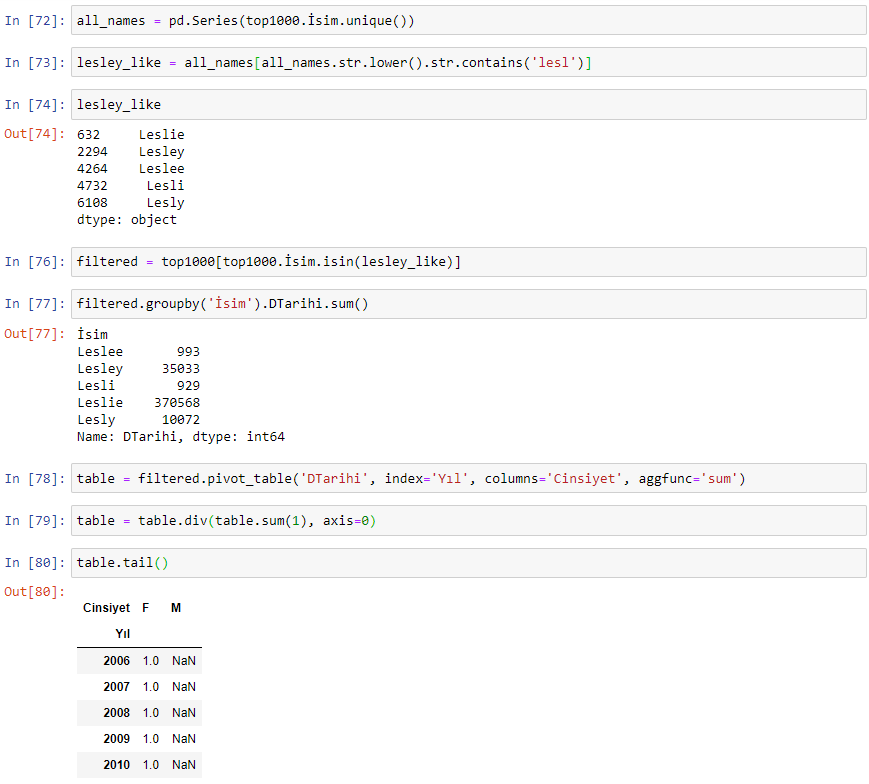


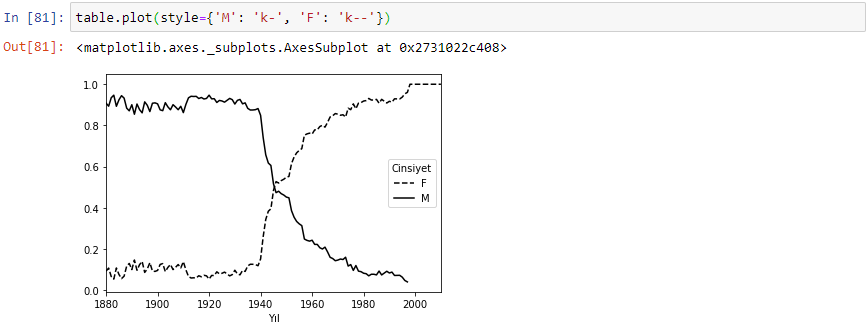






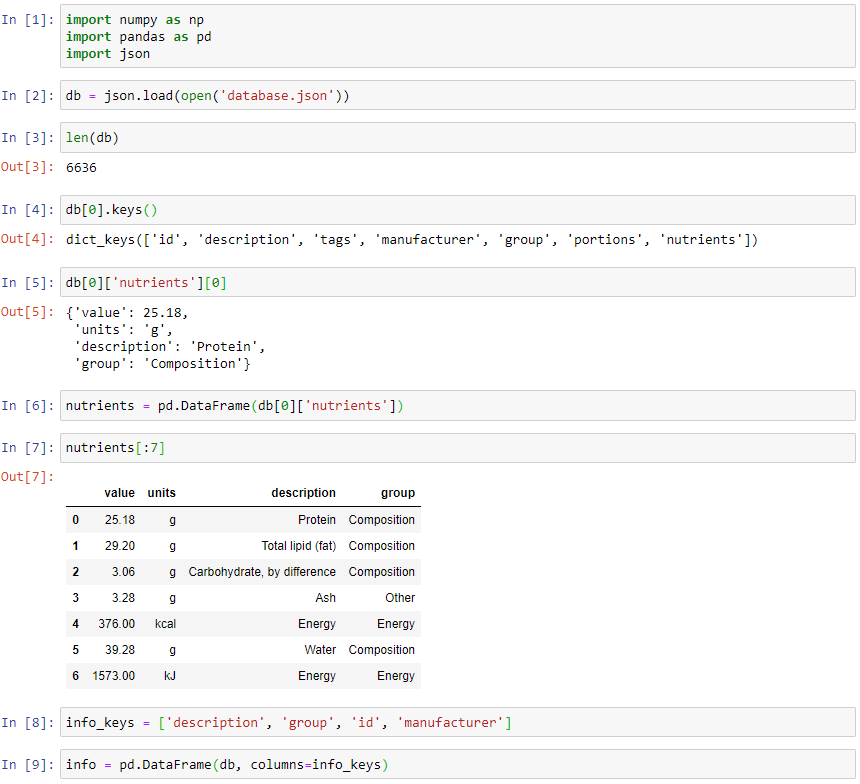
**Kız isimleri haline gelen erkek isimleri (ve tersi)**

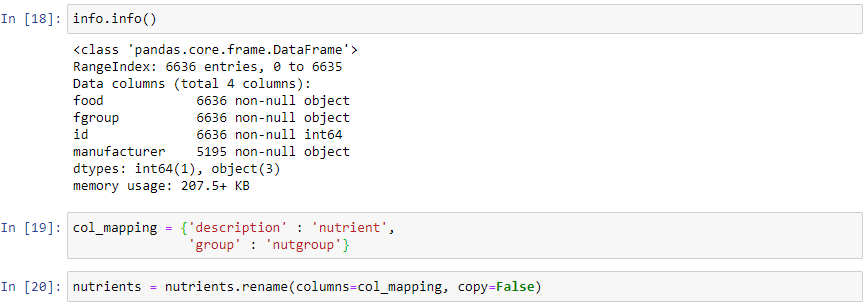
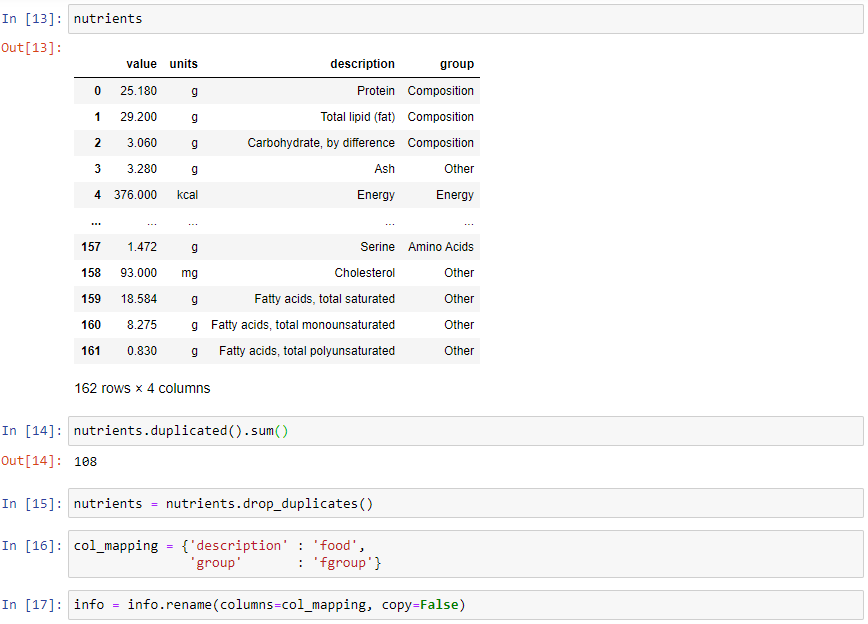




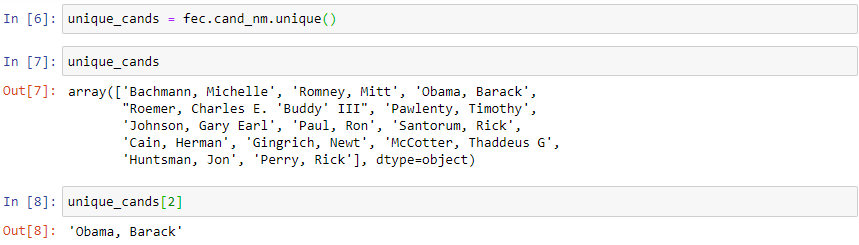
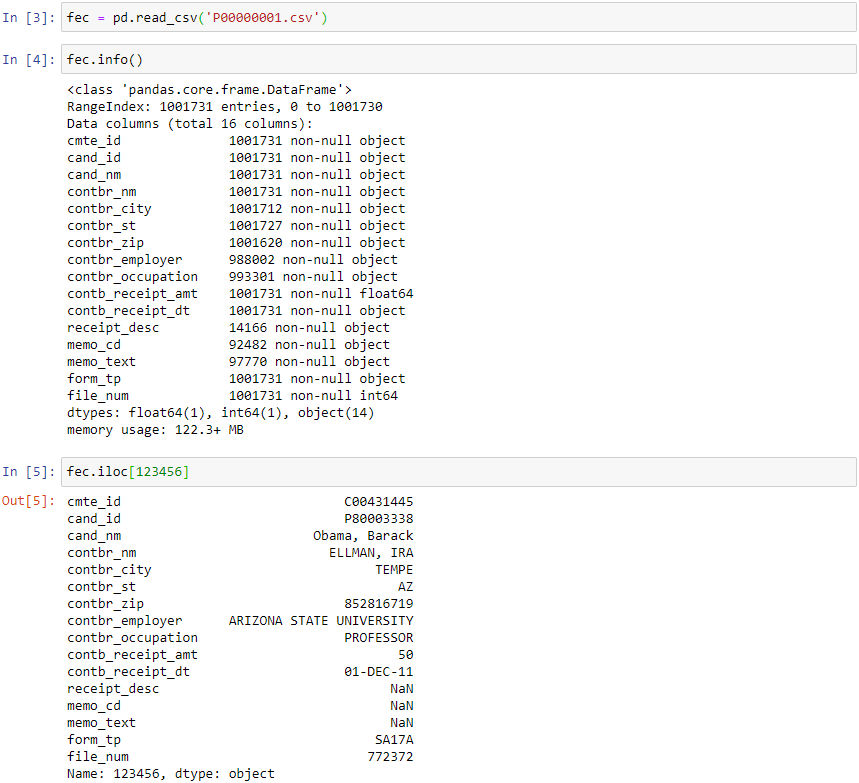
**14.4. USDA Gıda Veritabanı**

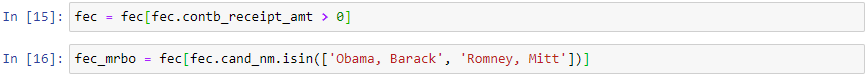
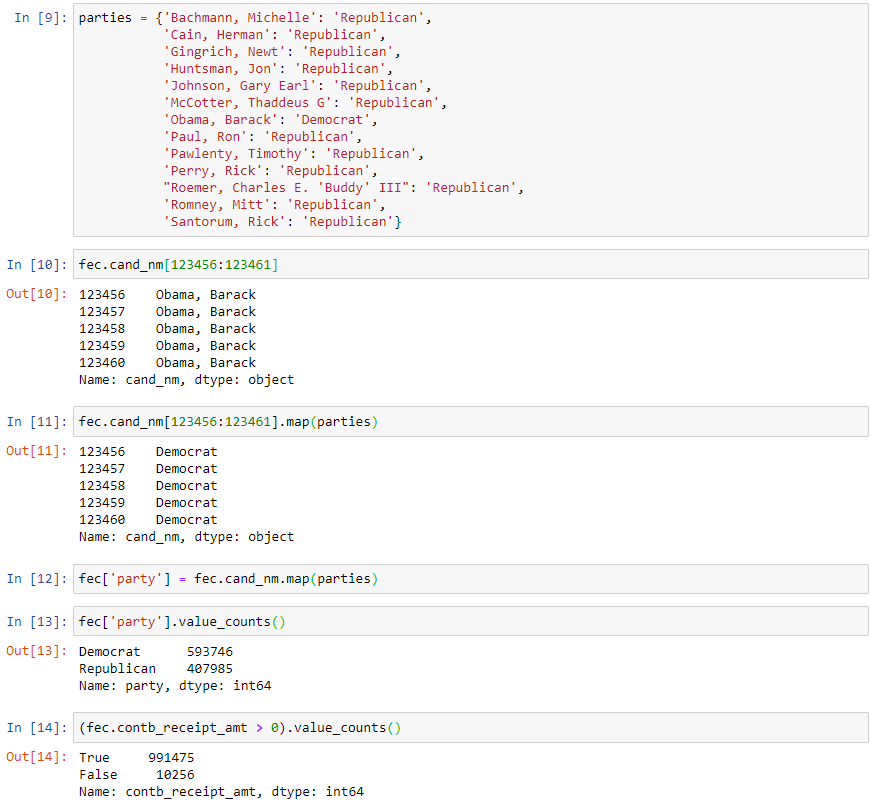
ABD Tarım Bakanlığı, gıda maddeleri bilgisi ile ilgili bir veri tabanı sunmaktadır. Programcı Ashley Williams, bu veritabanının bir sürümünü JSON formatında kullanıma sunmuştur.



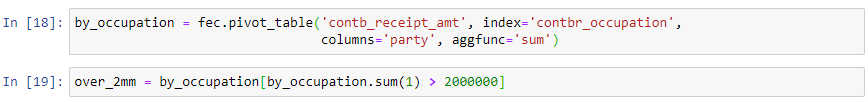
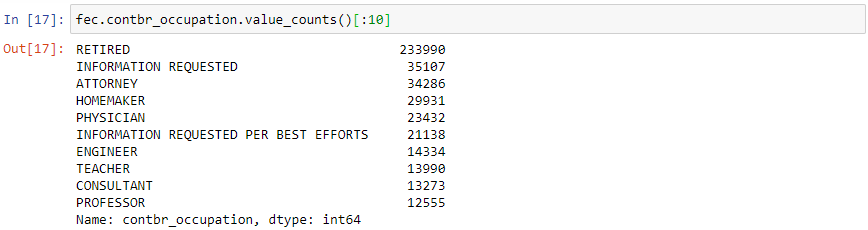


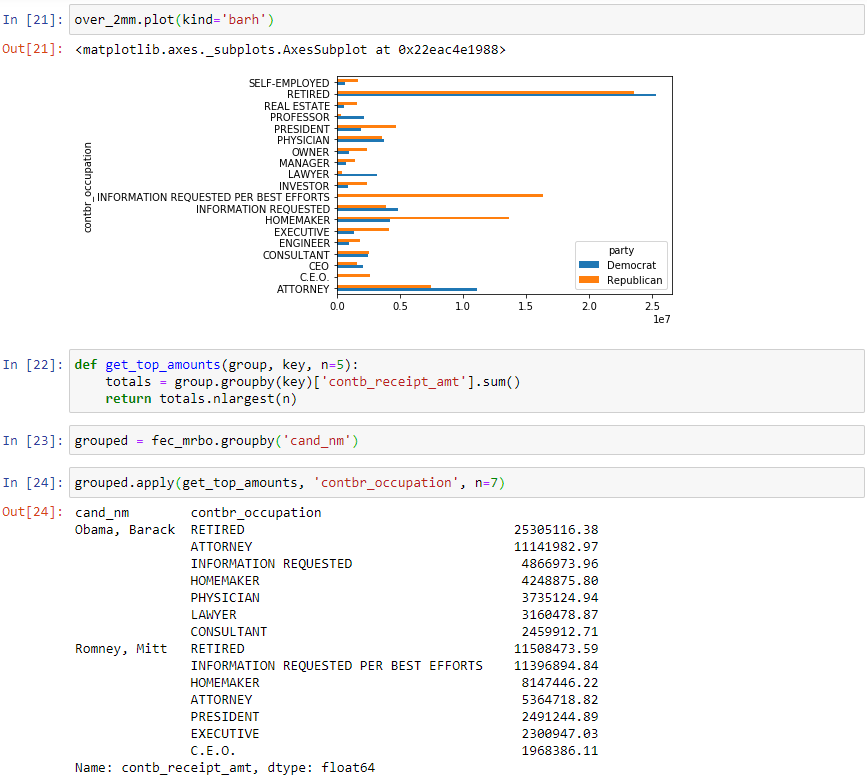
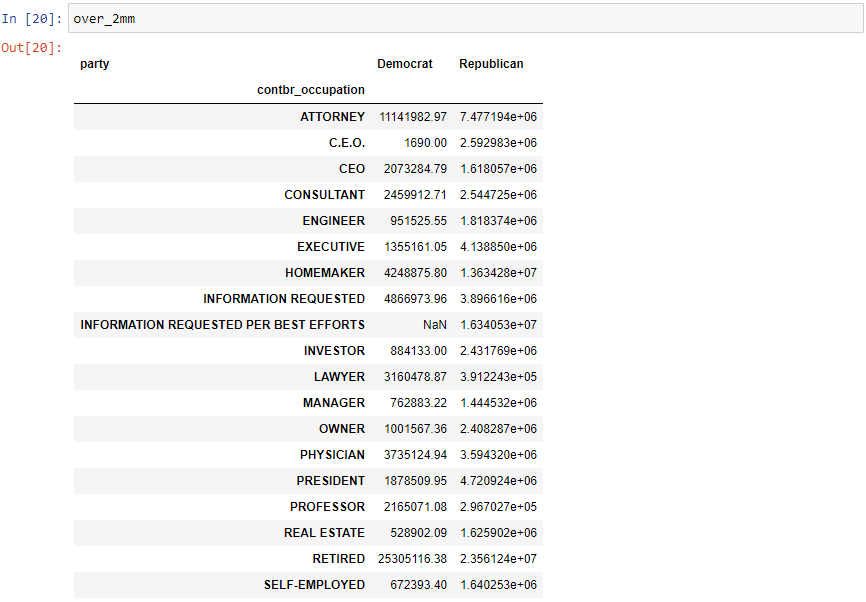
**14.5. 2012 Federal Seçim Komisyonu Veri Tabanı**

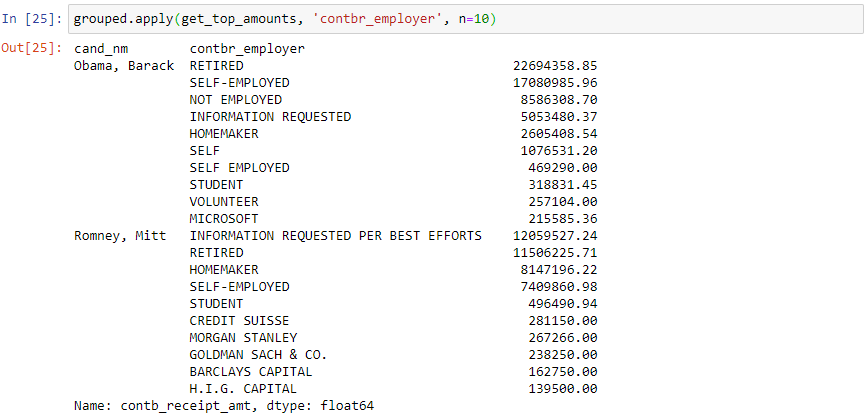




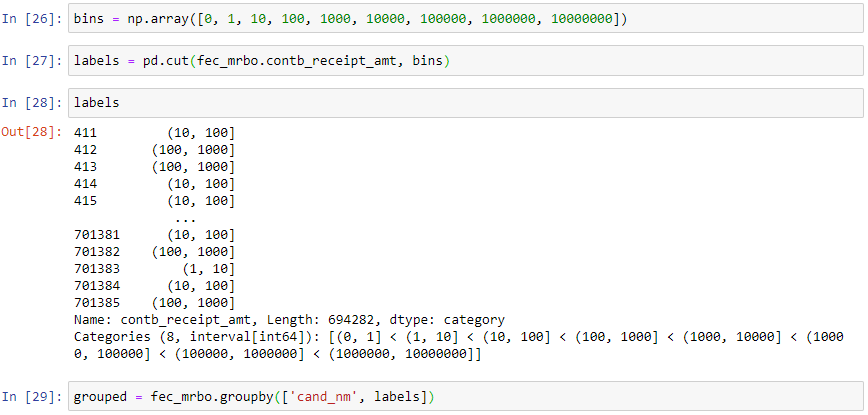
**Meslek ve İşveren Tarafından Bağış İstatistikleri**

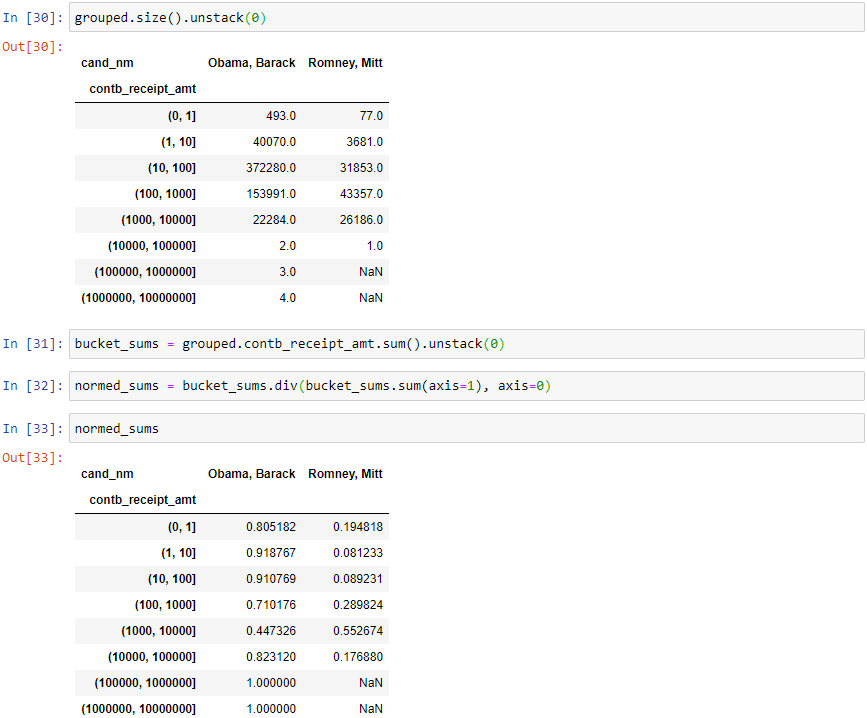


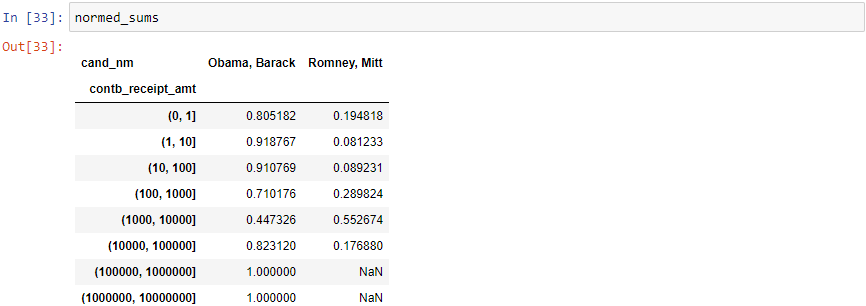




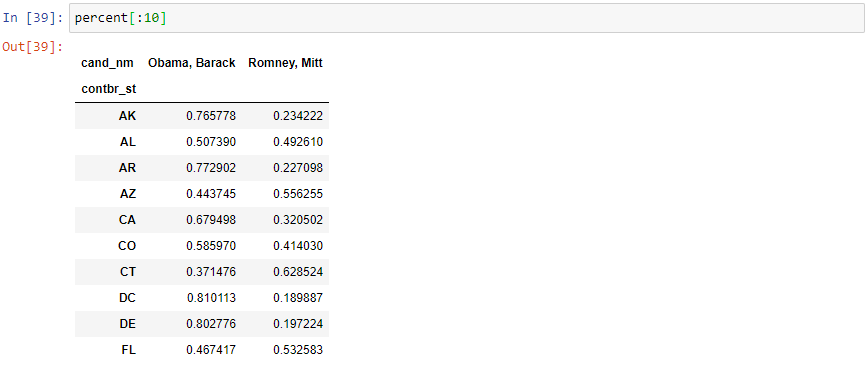
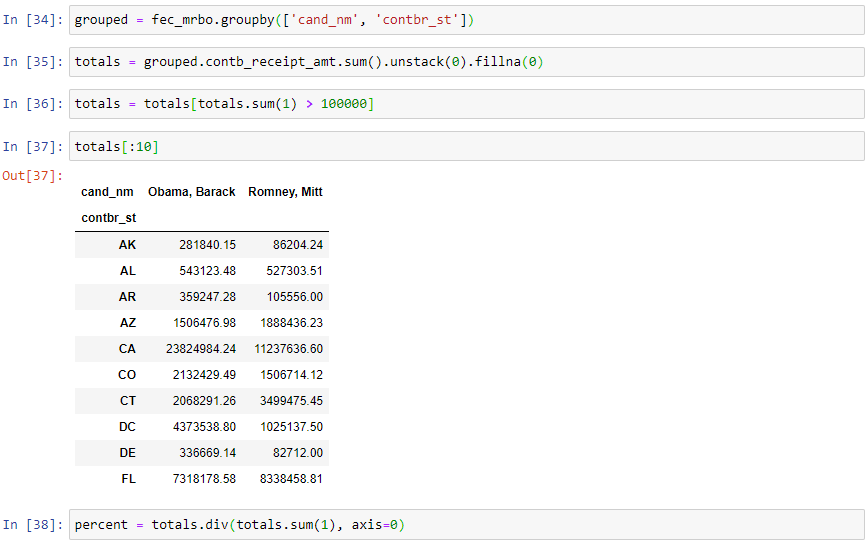
**Bucketing Bağışı Miktarları**







**Devlete Göre Bağış İstatistikleri**



**EK A: Gelişmiş NumPy**

Bu ekte, dizi hesaplama için NumPy kütüphanesinin derinliklerine ineceğiz. Bu, ndarray tipi hakkında daha fazla iç detay ve daha gelişmiş dizi manipülasyonları ve algoritmaları içerecektir.

**A.1. ndarray Nesne İçerikleri**

NumPy ndarray, çok boyutlu bir dizi nesnesi olarak homojen veri bloğunu (bitişik veya aralıklı) yorumlamak için bir araç sağlar. Veri türü veya dtype, verilerin kayan nokta, tam sayı, boolean veya baktığımız diğer türlerden biri olarak nasıl yorumlandığını belirler.

Ndarray'i esnek kılan şeyin bir kısmı, her dizi nesnesinin bir veri bloğu üzerinde basamaklı bir görünüm olmasıdır. Örneğin dizi görünümünün [:: 2, :: - 1] herhangi bir veriyi nasıl kopyalamadığını merak edebilirsiniz. Bunun nedeni, ndarray'ın sadece bir yığın bellek ve bir türden ibaret olmaması; aynı zamanda dizinin değişen adım boyutlarında hafızaya girmesini sağlayan “adım atma” bilgisine sahiptir. Daha doğrusu, ndarray dahili olarak aşağıdakilerden oluşur:

• Verilere bir işaretçi - yani, RAM’de veya bellek eşlemeli bir dosyada bir veri bloğu

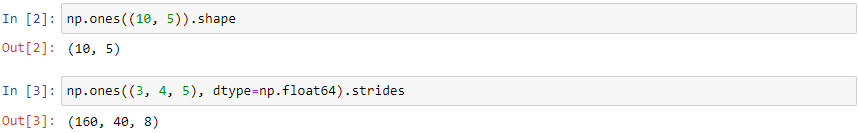
• Dizideki sabit boyutlu değer hücrelerini tanımlayan veri türü veya dtype

• Dizinin şeklini gösteren bir demet

• Bir boyut boyunca bir elemanı ilerletmek için bayt sayısını "adım" olarak gösteren tam sayılar dizisi



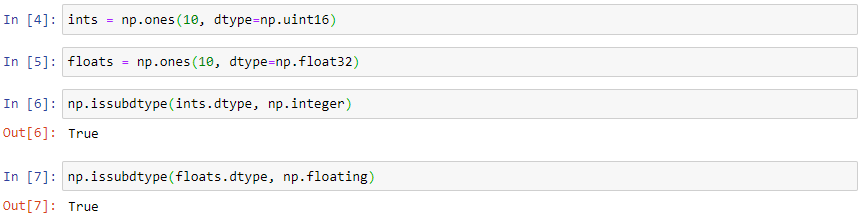
Örneğin, 10 × 5 dizisi (10, 5):



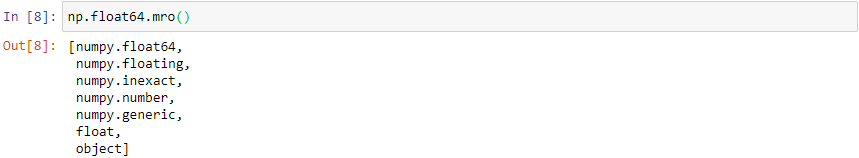
Tipik bir NumPy kullanıcısının dizi adımlarıyla ilgilenmesi nadir olsa da, “sıfır kopya” dizi görünümlerinin oluşturulmasında kritik bileşenlerdir. Adımlar bile negatif olabilir, bu da bir dizinin bellekten “geriye” hareket etmesini sağlar (örneğin, obj[::-1] veya obj[:, :: -1] gibi bir dilimde durum böyle olur).

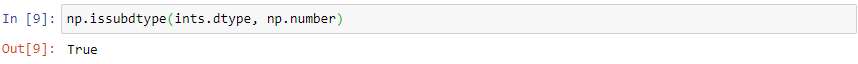
**NumPy dtype Hiyerarşi**

Zaman zaman bir dizinin integer, kayan noktalı sayı, string veya Python nesneleri içerip içermediğini kontrol edilmesi gereken kodunuz olabilir. Birden fazla kayan noktalı sayı tipi olduğu için (float16 ile float128 arasında), türün bir tür listesi arasında olup olmadığını kontrol etmek çok ayrıntılı olacaktır. Neyse ki dtypes, *np.issubdtype* işleviyle birlikte kullanılabilecek *np.integer* ve *np.floating* gibi süper sınıflara sahiptir.



Türün mro yöntemini çağırarak, belirli bir türün ana sınıflarının tümünü görebiliriz.

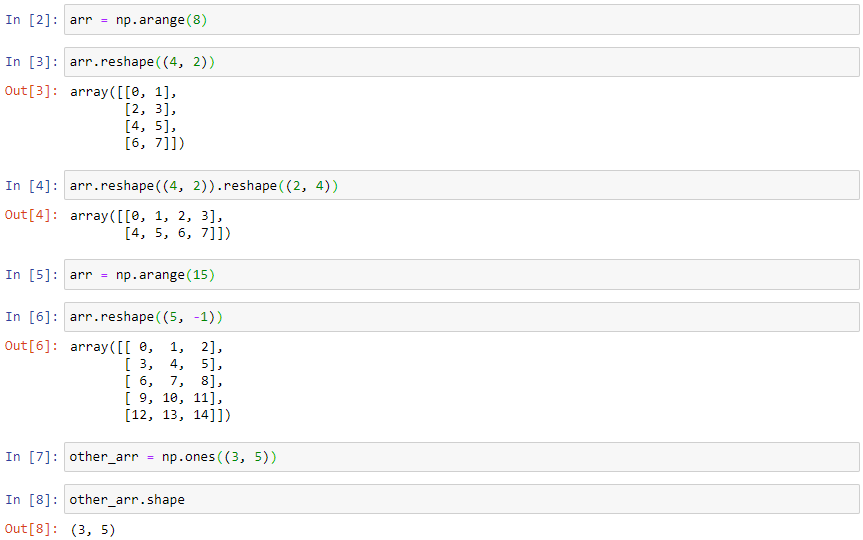


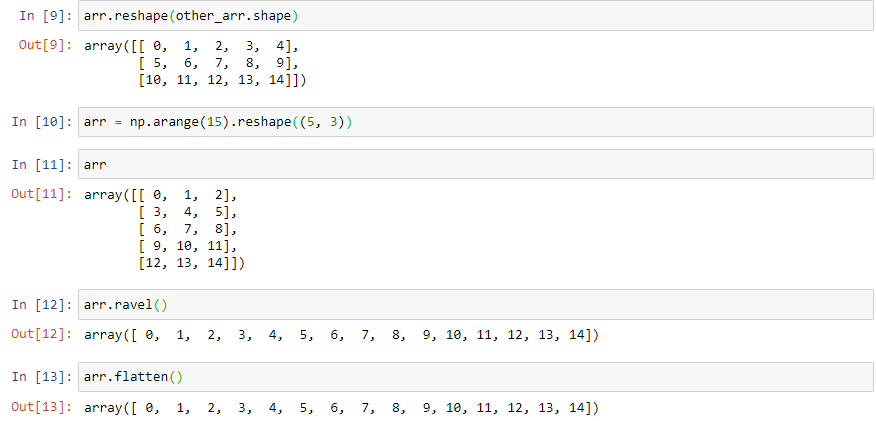


**A.2. Gelişmiş Dizi Manipülasyonu**

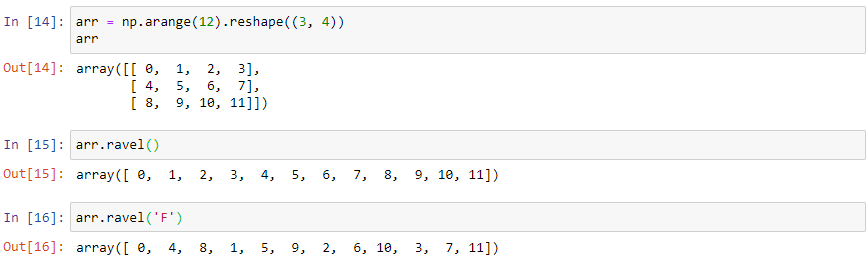
Süslü indeksleme, dilimleme ve boolean altkümesinin ötesinde dizilerle çalışmanın birçok yolu vardır. Veri analizi uygulamaları için yapılan yüksek düzey fonksiyonların çoğu pandalardaki üst düzey işlevlerle gerçekleştirilirken, bir noktada mevcut kütüphanelerden birinde bulunmayan bir veri algoritması yazmanız gerekebilir.

**Dizileri Yeniden Şekillendirme**

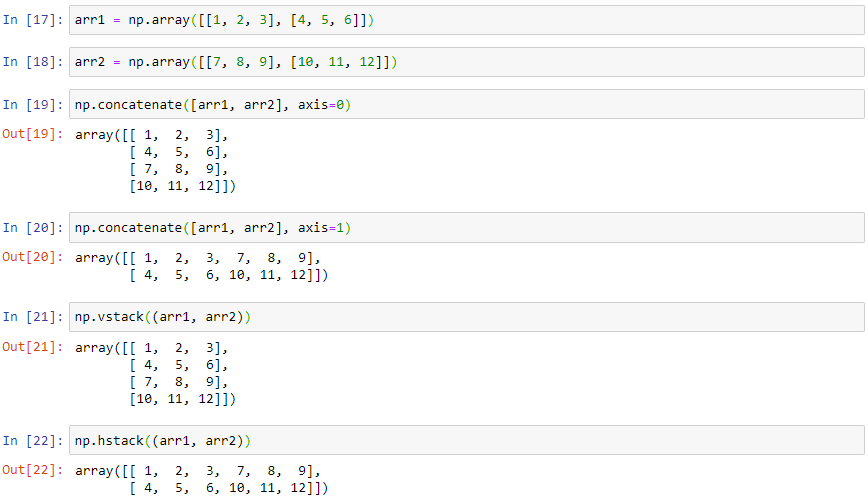


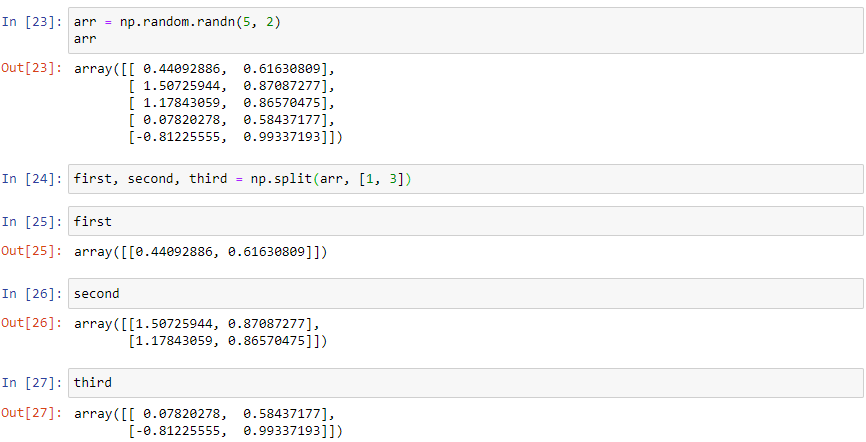


**C Versus Fortran Order**

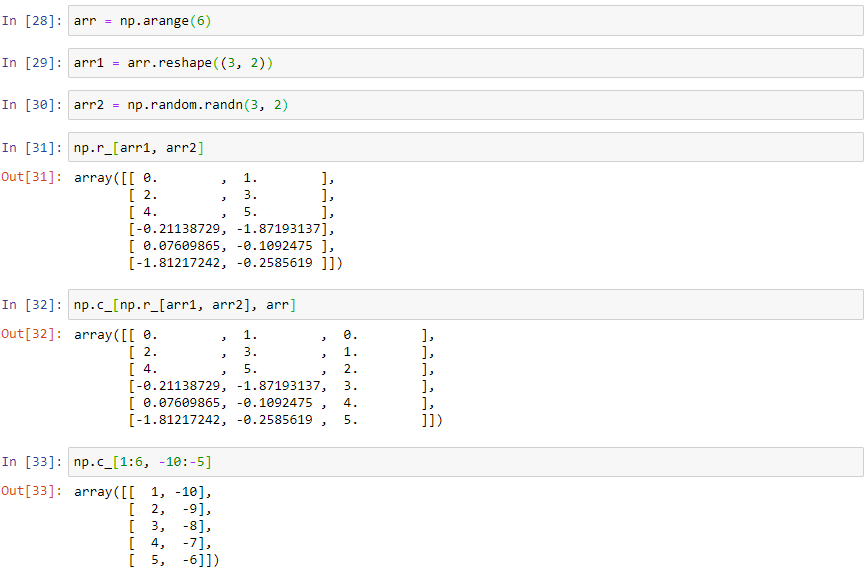


**Dizileri Birleştirme ve Bölme**

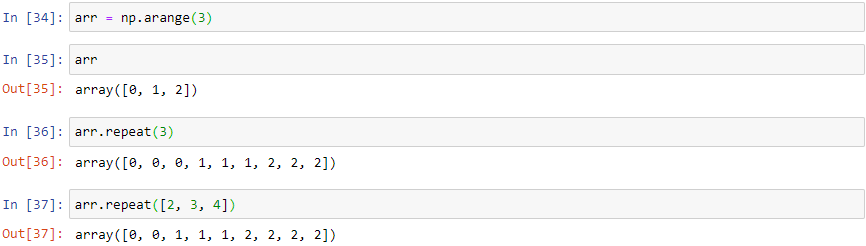


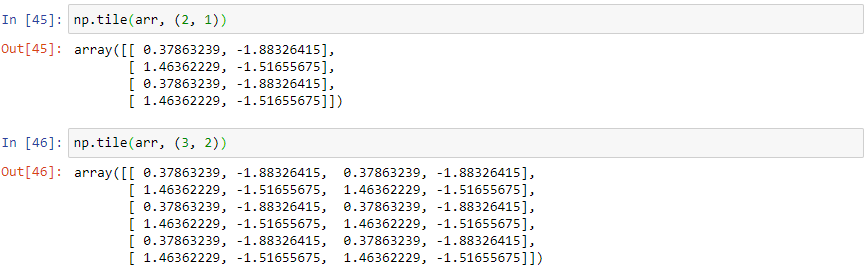
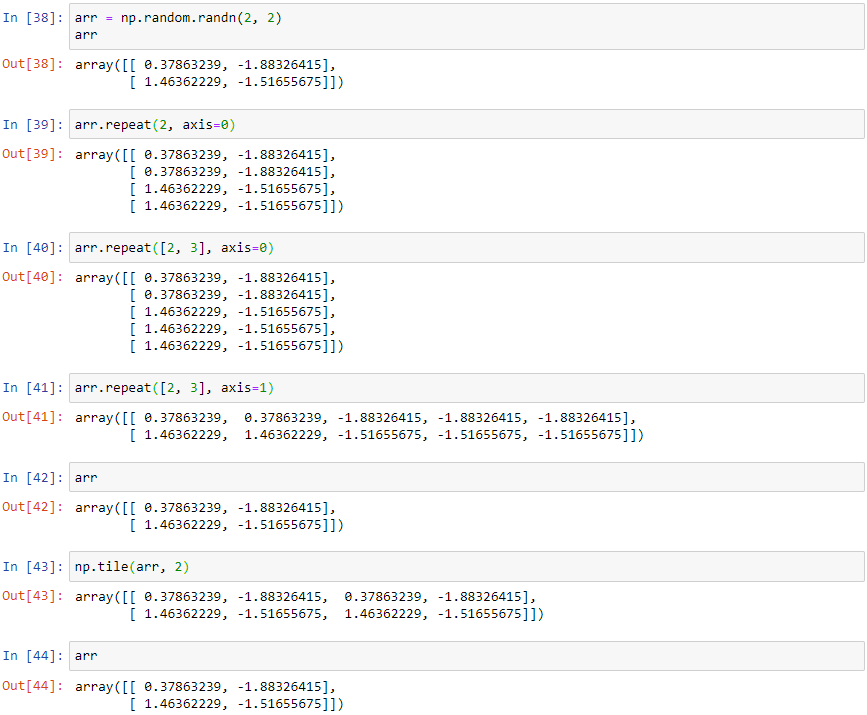


**İstifleme yardımcıları: r\_ ve c\_**

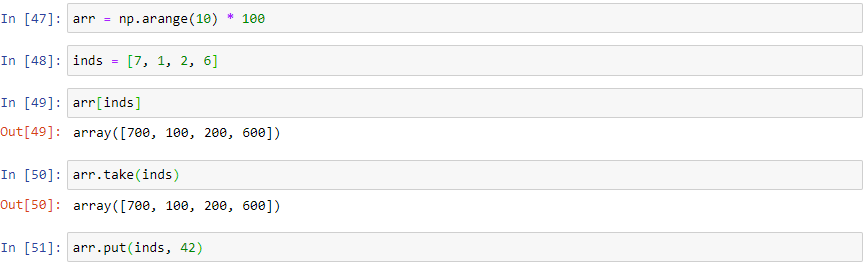


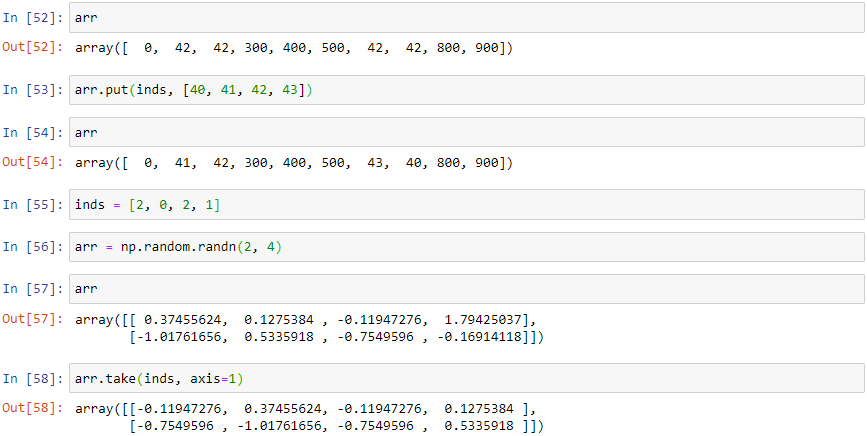
**Yinelenen Öğeler: Tile ve Repeat**





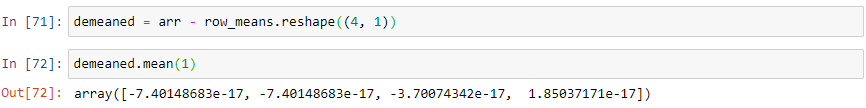
**Fantezi Endeksleme Eşdeğerleri: Take ve Put**



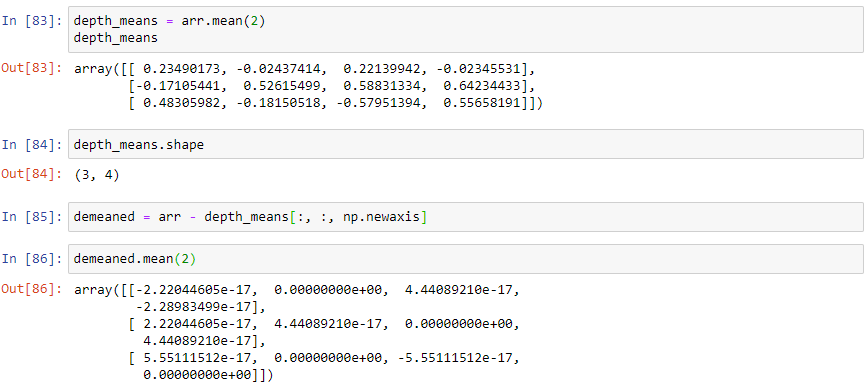
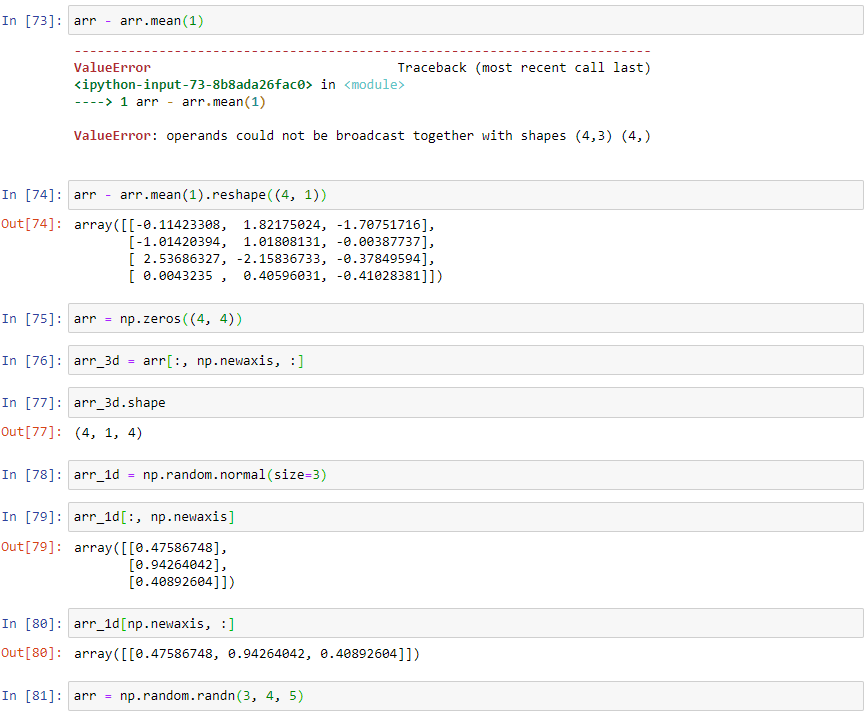


**A.3. Yayıncılık**

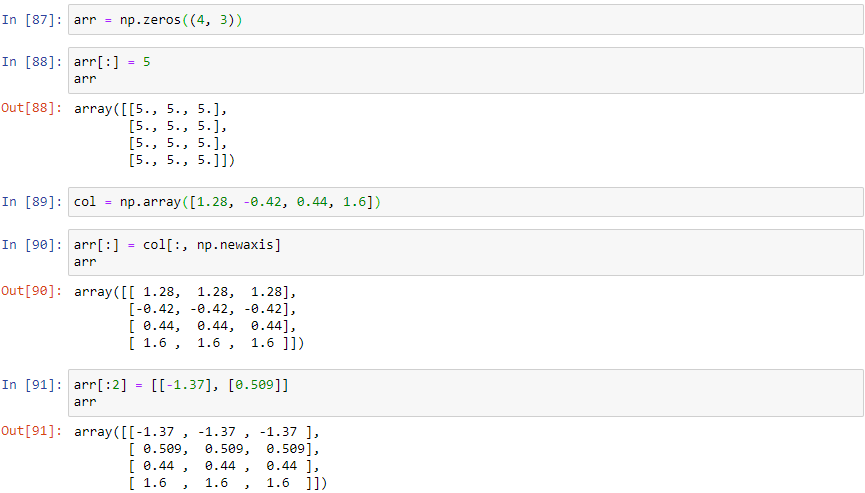




**Diğer Eksenlerde Yayın Yapma**

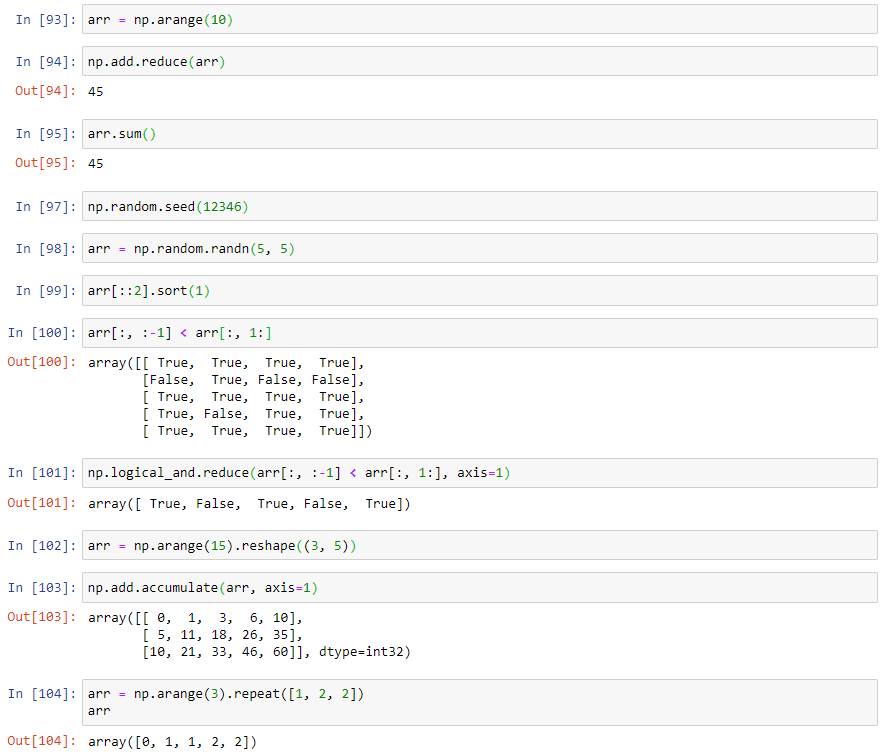


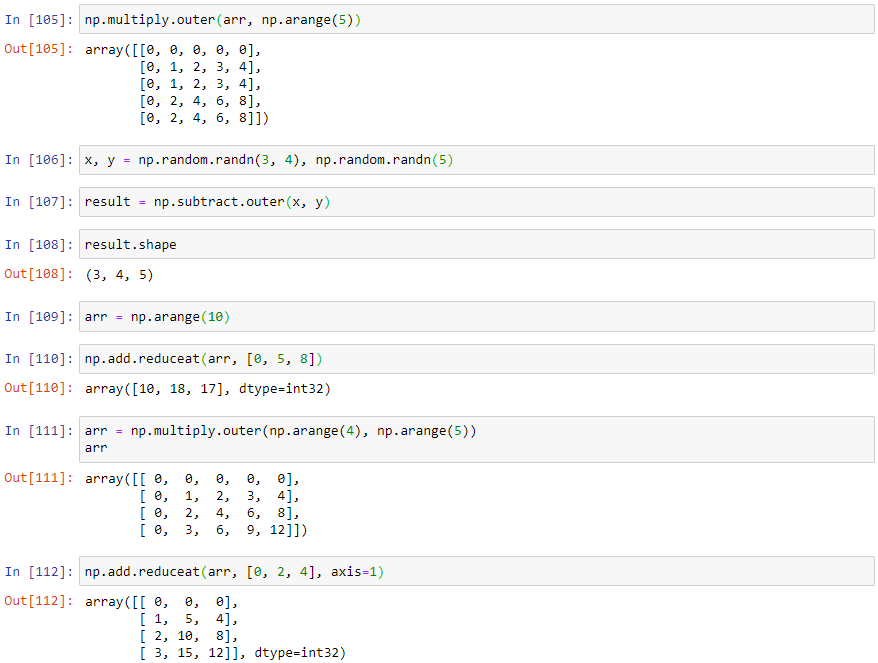
**Yayına Göre Dizi Değerlerini Ayarlama**



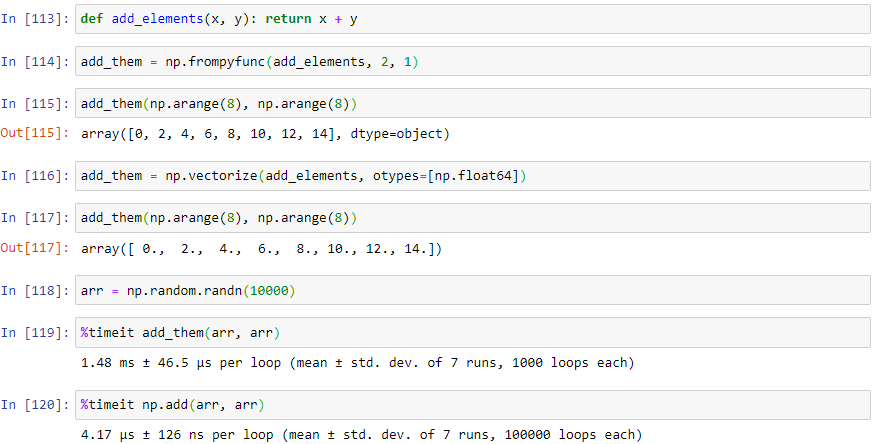
**A.4. Gelişmiş ufunc Kullanımı**

**ufunc Örnek Yöntemleri**

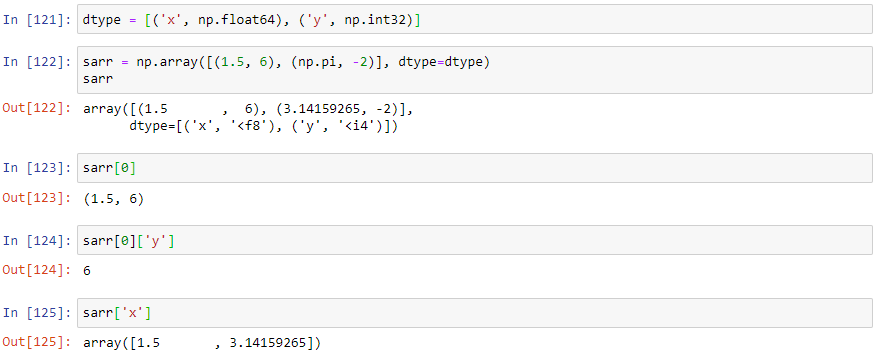




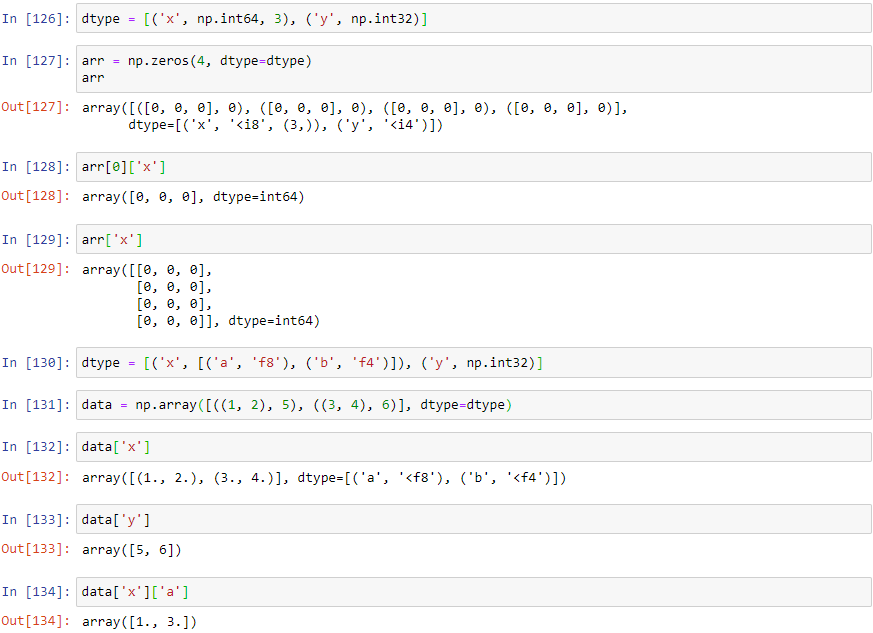
**Python'da Yeni ufuncs Yazma**



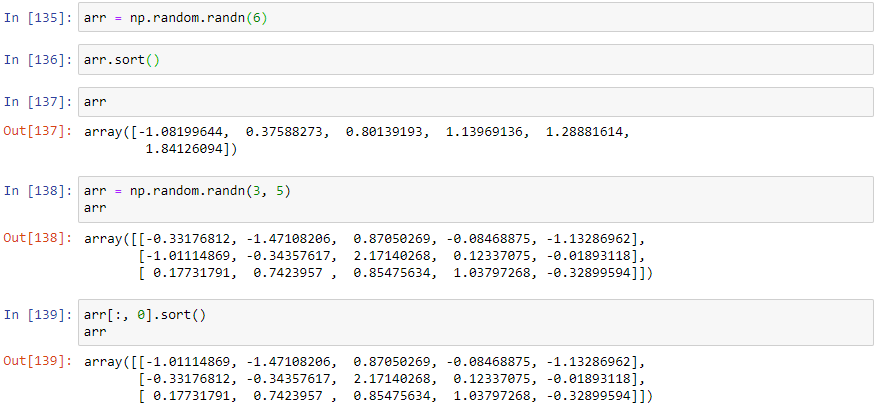
**A.5. Yapısal ve Kayıt Dizileri**

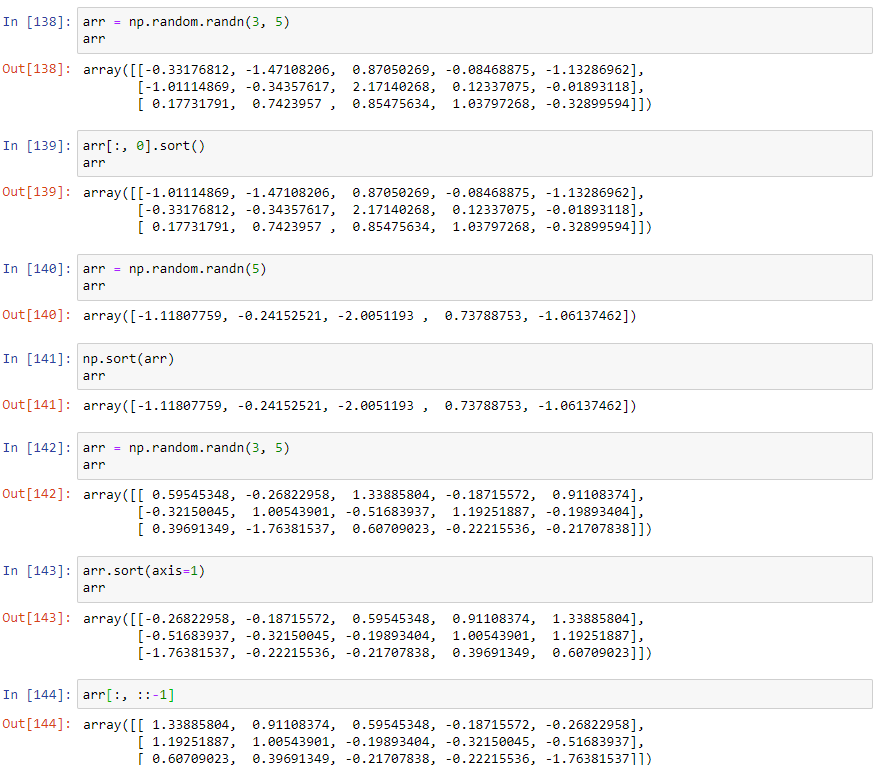


**İç İçe Geçmiş Türler ve Çok Boyutlu Alanlar**

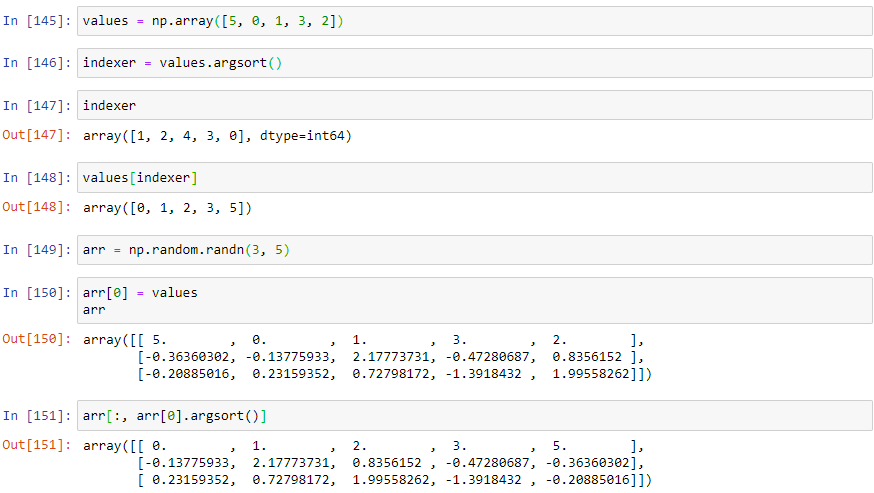


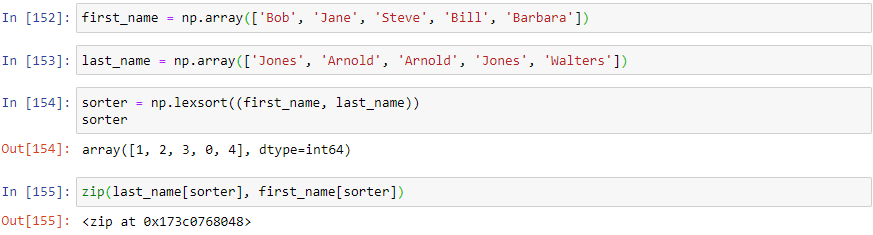
**A.6. Sıralama Hakkında Daha Fazla Bilgi**



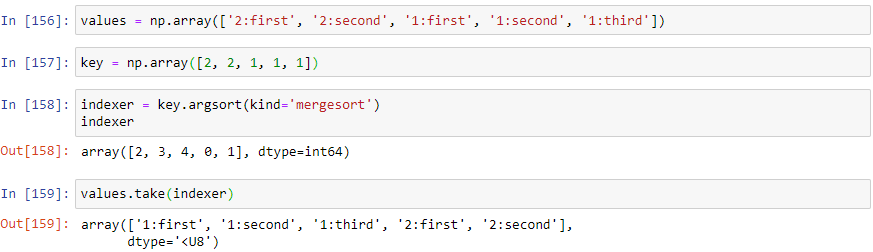


**Dolaylı Türler: argsort ve lexsort**

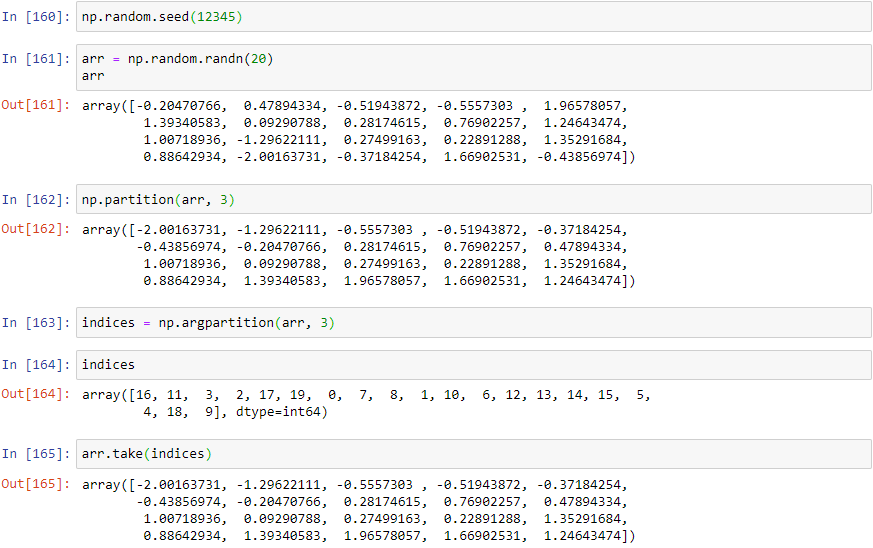




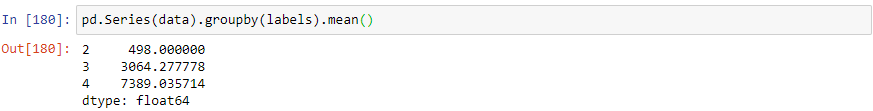
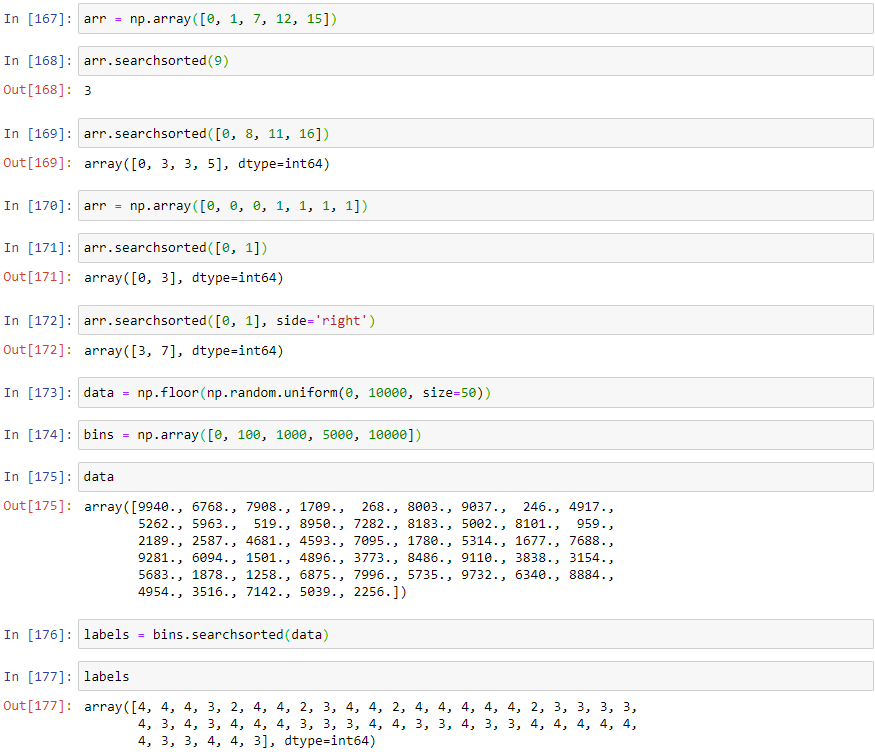
**Alternatif Sıralama Algoritmaları**



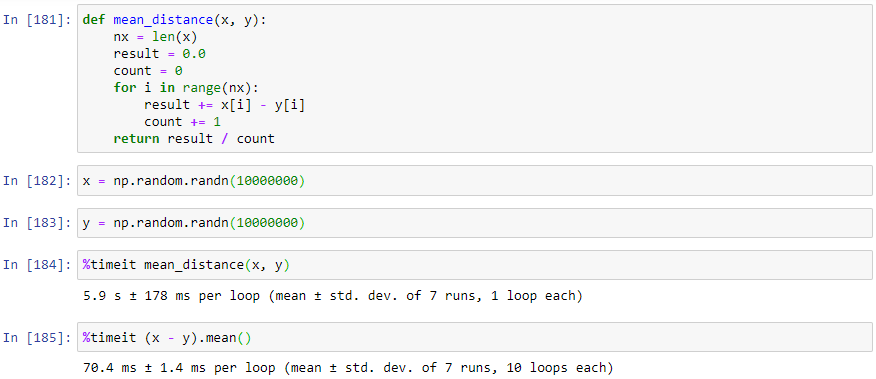
**Kısmen Sıralama Dizileri**

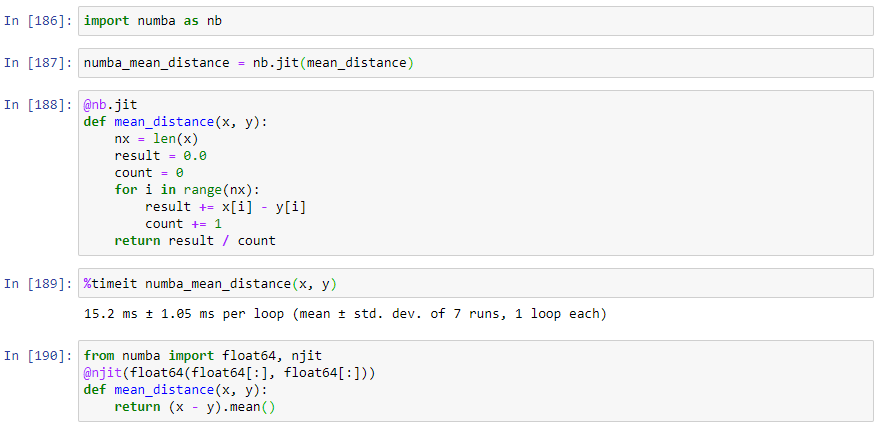


**numpy.searchsorted: Sıralanan Bir Dizideki Öğeleri Bulma**

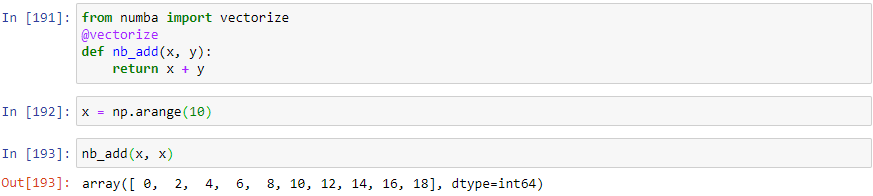


**A.7. Numba ile Hızlı NumPy İşlevlerini Yazma**



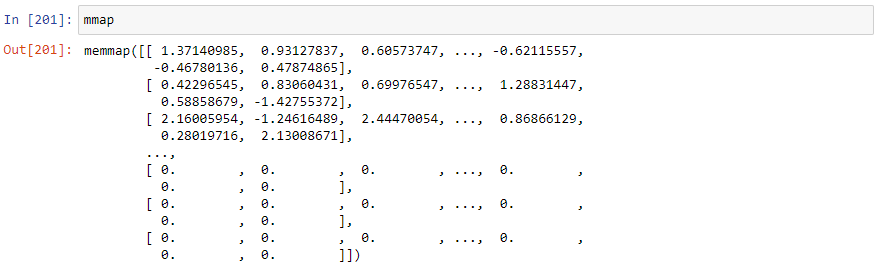
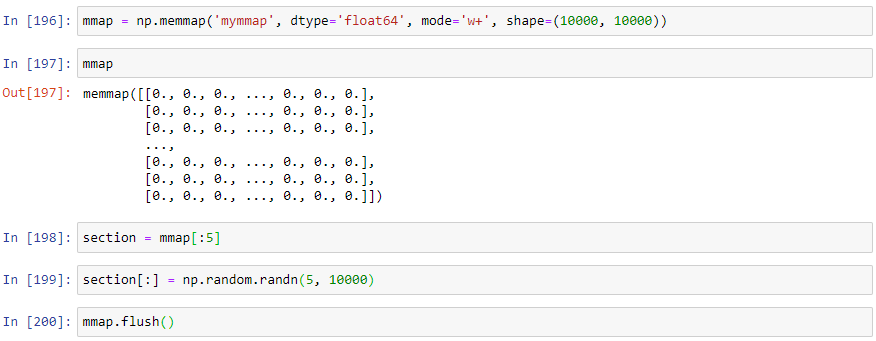


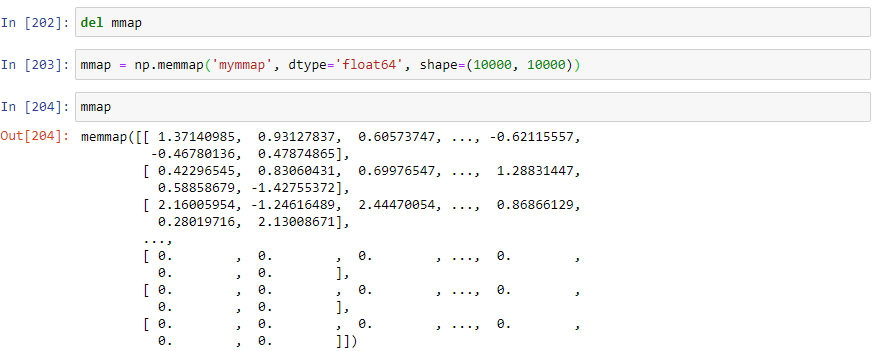
**Numba ile Özel numpy.ufunc Nesneleri Oluşturma**



**A.8. Gelişmiş Dizi Giriş ve Çıkışı**

**Bellek Eşlenmiş Dosyalar**





**A.9. Performans İpuçları**



