



Bölüm 1

1 - Başlangıç Bilgileri

c1t2-Sage Uygulama Sayfası

Semboller, Kümeler ve Sayı Sistemleri Tanıtımı Uygulamaları

Sage, yeni ve çok kapsamlı bir CAS uygulamasıdır. Sage University of Washington da matematik profesörü William Stein tarafından, 1905 de yayınlanmaya başlamıştır. Bugün, stabil 6.9 ve 6.10 beta sürümlerine erişilmiştir. Geniş bir gönüllüler grubu tarafından geliştirilmekte ve ücretsiz olarak dağıtılmaktadır.

Sage, diğer CAS sistemleri gibi kendi betik (scriting) program dilini değil, açık kaynak Python programlama dilini kullanmaktadır. Python geliştiricilerinin bazıları daha önce Ada programlama dili projesinde çalışmışlar ve bu dilin önemli özelliklerini Python programlama dilinin içeriğine katmışlardır. Bu nedenle, Python güncel programlama platformunun matematik uygulamaları açısından en önemli açık kaynak program dillerinden bir haline gelmiştir.

Sage, Python kitaplıkları yanında, birçok gelişmiş kitaplıkları (özgün programları) içermekte ve kullanıcıya erişim olanakları sunmaktadır.

Sage hernekadar Python program diline dayanmakta ise, bu dile birçok katkı sağlayan özel fonksiyonları da içermektedir. Bu nedenle, Sage yine de kendi özel program dilini oluşturmuş olduğu söylenebilir.

Sage, Linux sistemlerine doğrudan indirilip yerleştirilebilir. Windows sistemlerine Oracle, VirtualBox aracılığı ile bir sanal makine olarak indirilebilir. Bu sanal makine ancak İngilizce klavye ile kullanılabilir. Bu nedenle, web üzerinden kullanımı çok daha rahat olmaktadır. Özellikle cloud.sagemath.com organize olduktan sonra, istemci makineler Sage kurulumuna hiçbir gerek kalmadığı belirtilmektedir.

Sage ana sunucusu <http://cloud.sagemath.com> üzerinden üyelik kabul etmektedir. Buna rağmen, dünyanın bir çok yerlerinde Sage sunucuları bulunmakta ve herbirine üye olunarak kullanılabilir.

Bir diğer kolay kullanımı, <https://sagecell.sagemath.org/> sayfasındaki tek Sage uygulama hücrelerinin kullanılmasıdır. Bu sayfa sadece deneme ve ve çok küçük uygulamalar için yeterlidir. Düzenli uygulamalar için Sage bulut servisine abone olarak kullanmak sağlık verilir.

Sage olağanüstü geniş, kullanımı kolay ve başlangıç düzeyinden en gelişkin düzeye kadar çalışanlara katkı sağlayan bir CAS sistemidir. Bu olağanüstü sistemi o derece yaygın ve güçlüdür ki, birçok matematik ders kitabı Sage uygulamaları içermektedir.

Python ve Kümeler

Python programlama dili, Kümeleri doğrudan destekleyen az sayıda program dillerinden biridir. Python küme uygulamaları, winPython notebookları kullanılarak, aşağıda görüldüğü şekilde gerçekleştirilmiştir.

jupyter sets Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [57]: A= set([12 , 16 , 33 , 74, 56 , 99])
```

```
In [58]: print(A)
```

```
{33, 99, 74, 12, 16, 56}
```

```
In [59]: B = set([12, 46, 25, 38, 33 , 56,105 , 103])
```

```
In [60]: print (B)
```

```
{33, 38, 103, 105, 12, 46, 56, 25}
```

Python programlama dilinde, küme işlemleri istenirse, standart imgeler (token), istenirse, python küme nesnelerinin (tanımlı python kümelerinin), öntanımlı matotları

Not: | imgesi matematikteki dışlamayan OR anlamındadır. Bu imge, "eleman ya A kümesinin bir elemanıdır veya B kümesinin bir elemanıdır veya her ikini birden elemanıdır" şeklinde okunmalıdır. Bu işlem küme birleşimi (union) işlemidir.

```
In [61]: A | B
```

```
Out[61]: {12, 16, 25, 33, 38, 46, 56, 74, 99, 103, 105}
```

```
In [62]: A.union(B)
```

```
Out[62]: {12, 16, 25, 33, 38, 46, 56, 74, 99, 103, 105}
```

Not: & imgesi matematikteki (ve) (AND) anlamındadır. Bu imge, oluşacak kümenin elemnalarının hem A kümesinde, hem de B kümesinde olmasını, yani kümenin ortak elemanlarından oluşması gerektiğini belirtmektedir. Bu işlem kümelerin kesişim işlemidir.

```
In [63]: A & B
```

```
Out[63]: {12, 33, 56}
```

```
In [64]: A.intersection(B)
```

```
Out[64]: {12, 33, 56}
```

Küme Farkı B evrensel küme olarak kabul edilirse, B de olan fakat A da olmayan elementleri gösterir. Bu farka A nın komplemanı (A nın B ye tümleyeni) a

verilir. ve A' ile gösterilir.

```
In [78]: B-A
Out[78]: {25, 38, 46, 103, 105}

In [80]: print("A = ", A)
          print("B = ", B)

A = {33, 99, 74, 12, 16, 56}
B = {33, 99, 38, 103, 105, 74, 12, 46, 16, 56, 25}
```

$A \setminus B$ kümesinin B kümesine tümleyeni (A 'nın komplemanı = A' anlamını taşır. Bu $B \setminus A$ fark kümesidir.

```
In [87]: A^B
Out[87]: {25, 38, 46, 103, 105}

In [76]: ComplementA = B.difference(A)
          print(ComplementA)

{46, 105, 25, 38, 103}

In [82]: BB = A.union(ComplementA)
          BB
Out[82]: {12, 16, 25, 33, 38, 46, 56, 74, 99, 103, 105}

In [83]: B == BB
Out[83]: True

In [88]: A | (A^B)
Out[88]: {12, 16, 25, 33, 38, 46, 56, 74, 99, 103, 105}

In [89]: B == A.union(B.difference(A))
Out[89]: True
```

```
In [90]: B == A | (A^B)
Out[90]: True
```

Yukarıdaki kodlardan görüldüğü gibi, Python programlama dilinin, kümelere verdiği destek, dikkate değer niteliktedir.

Sage, bütün bu kodları aynen destekler. Bunun dışında Sage programlama dili, Python'dan daima daha fazlasını sunar.

Sage Kullanımı

Sage sisteminin kullanılması en kolay şekilde, <http://cloud.sagemath.org> sitesinde bir kullanıcı hesabı açtırmakla gerçekleşir. Hesap açımı ücretsizdir, fakat olanağı olanların aylık abone sistemine geçmeleri çok yararlı olmaktadır.

Sage aynı Python not defteri (notebook) arayüzü gibi çalışır. Her Sage hücresi, diğer hücrelerden bağımsız olarak çalışabilir (independly executable) bir hücredir ve aynen Mathematica CAS sistemi not defterlerinde olduğu gibi, CTRL+ENTER veya sayfadaki Run düğmesine basılınca çalışır. Daha önce başka hücrelerde tanımlanmış ve çalıştırılarak belleğe alınmış olan tüm tanımlar, daha sonraki hücrelerde de geçerli olurlar.

Her Sage hücresi otonom (özerk) bir hücredir. Çalıştırıldıklarında, bünyelerindeki her yasal bildirimi (statement) çalıştırlar. Bir değeri görüntülemek için değerin tanımlayıcısını (identifier), yani daha anlaşılır bir söylem ile değişken ismi, tek bir satıra veya aynı satıra nokta virgülden sonra yazılabilir veya show() veya print() öntanımlı fonksiyonlarından yararlanılabilir. Öntanımlı print() fonksiyonu, aynen Python programlama dilinde olduğu gibi çalışır.

Her Sage hücresinin belirli bir meta tanımlayıcısı vardır. Hiçbir meta bilgisi verilmezse, bu hücre bir hesaplama hücresi olarak işlev görür. Eğer hücrenin ilk satırına,

```
%html
```

meta tanımlayıcısı yazılırsa bu hücre bir html kod sayfası gibi çalışır. Çalıştığında yazılı html kodlarını aynen bir kod çözümleyicisi gibi (örnek olarak Internet Explorer, Edge veya FireFox gibi) çözümleyerek görüntüler.

Eğer hücrenin ilk satırına,

```
%md
```

meta tanımlayıcısı yazılırsa bu hücre bir markdown kod sayfası gibi çalışır. Çalıştığında yazılı metin kodlarını aynen bir metin kod çözümleyicisi gibi (örnek olarak Notepad, Scite gibi) çözümleyerek görüntüler. Böylece, açıklama metinleri, yorumlar ve benzerleri, sayfaya kolayca eklenebilirler.

Eğer hücrenin ilk satırına,

```
%md
```

meta tanımlayıcısı yazılırsa bu hücre bir markdown kod sayfası gibi çalışır. Çalıştığında yazılı metin kodlarını aynen bir metin kod çözümleyicisi gibi (örnek olarak Notepad, Scite gibi) çözümleyerek görüntüler. Böylece, açıklama metinleri, yorumlar ve benzerleri, sayfaya kolayca eklenebilirler.

Eğer hücrenin ilk satırına,

```
%auto veya salvus.auto(True)
```

meta tanımlayıcılarından birisi yazılırsa bu hücre bir başlangıç (initialisation) hücresi gibi çalışır. Başlangıç hücreleri, not defterinin her yüklenmesinde otomatik olarak çalışır. Bu şekilde, bir başlangıç hücresinde, her türlü kullanıcı sabitleri, tüm sayfa boyunca kullanılacak tanımlar, global değişkenler gibi değerler, bir başlangıç hücresinde tanımlanarak, sayfanın her yüklenmesinde tanımlı olmaları sağlanabilir.

Başlangıç hücreleri sayfa başında olabildikleri gibi, her Sage hücresine de otomatik çalışma özelliği kazandırılabilir. Bir Sage hücresinin meta bilgileri bu hücrenin en az bir kez çalıştırılması ile aktif hale geçirilmiş olurlar.

Bu sayfada görülen Sage kodları, <http://bedriemir.com/algebra/c1/t2/uygulama-1/sage/c1t1-uygulama-1-sage-kodlar.html> sayfasından erişilebilir. Buradaki kodlar kopyalanarak kendi açtığınız Sage not defteri sayfasında kullanılabilir. Alternatif olarak, <http://bedriemir.com/algebra/c1/t2/uygulama-1/sage/c1t1-uygulama-1-sage.sws> dosyası kendi hesabınıza yüklenebilir (upload)

Sage, küme tanımları kendi öz anahtar sözcüğü (keyword) veya kendi öntanımlı fonksiyonu veya bildirimi (statement) Set() fonksiyonu ile yapılabilir. Sage Set() fonksiyonu ile küme tanımlanması, aşağıdaki gibi yapılabilir.

Sage Küme Tanımları

Önce, bir Python listesi (list) veya bir Python topluluğu (tuple) veya bir Python kümesi (set) yaratılır.

```
Liste = [1, 2, 3]
Topluluk = (1, 2, 3)
Kume = {1, 2, 3}
```

Burada, istenilen veri tipi yaratılabilir. Fakat, değişmez (immutable) bir veri tipi olan topluluk (tuple) veri tipi çok elverişli olmaktadır.

Daha sonra, uygun Python veri tipinden, Sage küme tipine geçiş yapılır.

```
A = Set(Topluluk)
```

Python küme tipi tanımı `set()` (küçük harf s) ile yapılırken, Sage küme tanımının `Set()` (büyük harf S) ile yapıldığına dikkat etmek gerekir.

İstenirse, ilk iki adım birlikte gerçekleştirilebilir. Yani, bir Python veri tipi tanımlanıp, aynı anda Sage kümesine dönüştürülebilir.

```
A = Set((12, 14, 2, 5))
```

Kısa ve anlaşılır olduğundan, bu tanımın uygulanması sağlık verilir.

Sage kümeleri sıralanması gelişigüzel (random) elemanlardan oluşur. Elemanlarının sıralı halinin görüntülenmesi için `sorted()` fonksiyonu çağrılır. Bu fonksiyon kümeyi sıralı listeye dönüştürerek görüntülenmesini sağlar. Aşağıdaki işlem, tahrip edici bir atama olmadığından, orijinal küme değişkeni etkilenmez.

```
sorted(A)
A
[2, 5, 12, 14]
{12, 14, 2, 5}
```

Sage küme nesneleri, öntanımlı olarak birçok metodu (nesne ile uygulanabilen fonksiyonları) kalıtım ile edinirler. Bu fonksiyolardan en çok kullanılanlar:

```
cardinality()
difference()
CartesianProduct()
intersection()
parent()
rename()
subsets()
symetric_difference
is_subset()
is_superset()
```

metotlarıdır.

```
A=Set([4,2,3,1])
A
{1, 2, 3, 4}

B = Set([3,4,5,6,7,8,9])
B
{3, 4, 5, 6, 7, 8, 9}

A.cardinality()
4

B.cardinality()
7
```

Bir kümenin kardinalliği element sayısına eşit gibi düşünülebilir, fakat anlamı çok daha derindir.

```
B-A
```

```
{8, 9, 5, 6, 7}
```

B kümesinde olup A kümesinde olmayan elementlerin kümesi. Bu kümeye A'nın B'ye tümleyicisi veya A'nın B'ye göre komplemanı adı verilir. B kümesi evrensel küme olarak düşünülebilir. Bu küme, Sage küme nesnelerinde, `difference()` metodu olarak tanımlanmıştır.

C

```
{8, 9, 5, 6, 7}
```

olarak bulunabilir. Yani eğer bir A kümesi Sage küme değişkeni olarak tanımlanmışsa, `A.difference(B)` deyimi, A kümesinin B'ye göre tümleyicisi olan kümeyi geri döndürür.

```
complementA = B.difference(A)
```

Sage hücrelerinde, atama yapılırken, atanan değer görüntülenmez, Atanan değer görüntülenmesi için, ya atamanın yapıldığı değişken adını, ya da `show()` veya Python `print()` fonksiyonundan yararlanmak gerekir.

```
complementA = B.difference(A)
complementA
```

```
{8, 9, 5, 6, 7}
```

Bir A Sage küme nesnesinin, bir B Sage küme nesnesi ile kesişim kümesi, `intersection()` metodu ile geri döndürülür.

```
A.intersection(B)
```

Sage union fonksiyonu sadece kümeler için değil, listeler, topluluklar ve kümeler için uygulanabilen bir kendi başına çalışan (stand alone) fonksiyondur. Önce `union()` fonksiyonun Sage tarafından açıklanan uygulama yöntemini görelim:

```
help(union)
```

```
Help on function union in module sage.misc.misc:
```

```
union(x, y=None)
```

```
Return the union of x and y, as a list. The resulting list need not
be sorted and can change from call to call.
```

```
INPUT:
```

```
- ``x`` - iterable
```

```
- ``y`` - iterable (may optionally omitted)
```

```
OUTPUT: list
```

```
EXAMPLES::
```

```
sage: answer = union([1,2,3,4], [5,6]); answer
[1, 2, 3, 4, 5, 6]
sage: union([1,2,3,4,5,6], [5,6]) == answer
True
sage: union((1,2,3,4,5,6), [5,6]) == answer
True
sage: union((1,2,3,4,5,6), set([5,6])) == answer
True
```

Bu açıklamaya göre `union()` öntanımlı fonksiyonu geriye bir liste döndürecektir. Bu listenin bir Sage kümesine dönüştürülmesi gerekmektedir. Örnek olarak, elemanları sırasız olarak tanımlanmış iki tane Sage küme nesnesi tanımlayalım ve birleşimlerini inceleyelim

```
QA = Set([2,6,3,6,8,1])
QB = Set([16,9, 8, 7, 23, 36, 18, 11])
WW = union(QA,QB)
WW

[1, 2, 3, 36, 6, 7, 8, 9, 11, 16, 18, 23]
```

Bu sonuçtan, Sage `union()` fonksiyonun kullanımı iyice aklımıza yerleşmiş olmalıdır.

Bir A kümesinin bir B kümesi ile kesişiminin, A kümesinin B kümesine tümleyicisi ile birleşimi, B kümesini verir.

```
B == Set(union(A.intersection(B) , B.difference(A)))

True
```

Burda, `union()` fonksiyonu geriye liste tipinde bir veri döndürdüğü için, B kümesi ile karşılaştırabilmek amacı ile, `union()` fonksiyonunun geri döndürdüğü veri tipini, Sage kümesi tipine dönüştürümü gerçekleştiriliyor. Bu işleme veri tipi eşleştirilmesi (*coercion*) adı verilir. Ancak aynı veri tiplerinden değerler, birbirleri ile karşılaştırılabilir.

Sage CAS sisteminin kümeleri gayet iyi desteklediği ve bu sistemle tüm küme işlemlerinin, Python programlama diline göre daha ileri düzeyde gerçekleştirilebileceği görülüyor. İleride, Sage CAS sistemi ile daha başka küme uygulamaları gerçekleştireceğiz.

Sage ile Sayısal İşlemler

Sage girilen veri tipini algılar ve bu veri tipinin çalışabileceği en yüksek düzeyde duyarlık sağlanabilecek türe dönüştürür.

```
range(20)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

range(12,28,2)

[12, 14, 16, 18, 20, 22, 24, 26]

QR = Set(range(1,11))
QR

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Kesin (exact) sayılar arasındaki işlemlerin sonucu kesin sayılardır.

```
1 + 1/2

3/2
```

Kesin sayılar ile gerçel sayılar arasındaki işlemlerin sonucu gerçel sayılardır.

```
1 + 0.1
```

```
1.1000000000000000
```

Asal sayılar

```
primes(50)
```

```
<generator object primes at 0x7f90914568c0>
```

```
list(primes(50))
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

```
AP=tuple(sorted((primes(10,80))))
AP
```

```
(11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79)
```

Asal sayı çarpanları

```
factor(5736)
```

```
2^3 * 3 * 239
```

Kesin sayı işlemlerinin sonucunun gerçel sayıya dönüştürülmesi (coercion)

```
QA=n((1+1/2))
QA
```

```
1.5000000000000000
```

Sage otomatik olarak oranları en küçük terimlerine indirger.

```
QP = 25/250
QP
```

```
1/10
```

Asal çarpanların bulunması

```
factor(4567443689)
```

```
6577 * 694457
```

Bu işemi bilgisayar olmadan yapabilmek, hiç de kolay sayılmaz!

En Büyük Ortak Bölen (EBOB) (greatest common divisor) (gcd)

```
gcd(76,86)
```


2

76/82

36/41

Sage otomatik olarak en düşük terimlere indirgediği için, EBOB bulunmasına gerek kalmamaktadır. Ayrıca artık el ile bölme pek yapılmadığından, EBOB bulunmasına gerek azalmıştır.

Sage çok geniş bir sayısal destek sağlamaktadır. İleride bu yöntemleri uygulayarak daha fazla bilgi kazanacağız. Bu uygulamaları web sayfanızda açacağınız, sws (sage worksheet) sayfanızda denemeniz çok yararlı olacaktır.

[« Matematik İndeksi](#) [« Konu Sayfasına Dönüş](#) [Site İndeksi »](#)



Created with MSExpressionWeb4 [pdf sürümü](#)

Belgenin Son Düzenlenme Tarihi : Mon Nov 09 2015 23:22:54 GMT+0200 (Turkey Standard Time)