

Jupyter Notebook Kullanımı

Jupyter notebook daha önce değinildiği üzere Anaconda ile gelen bir uygulamadır. Özelliği ise hem hatırlatma notları yazılabilmesi hem de Python ile kod yazılabilmesidir. Başlangıç safhasında yani Python programlama dilini öğrenmeye yeni başlamışken hatırlatma notlarını tutmak pratik olacaktır.

Jupyter Notebook' u Anaconda Navigator'dan Launch diyerek ya da (kurulum aşamasında sistem pathine eklenmesi seçilmişti) Windows terminalinden Jupyter Notebook yazarak çağırabiliriz. Anımsanacağı üzere Jupyter Notebook web arayüzü ile açılacaktır ve beraberinde de bir terminal açılır. Bu terminalin, Jupyter notebook ile yapılan çalışma tamamlanıncaya kadar kapatılmaması gerekmektedir!

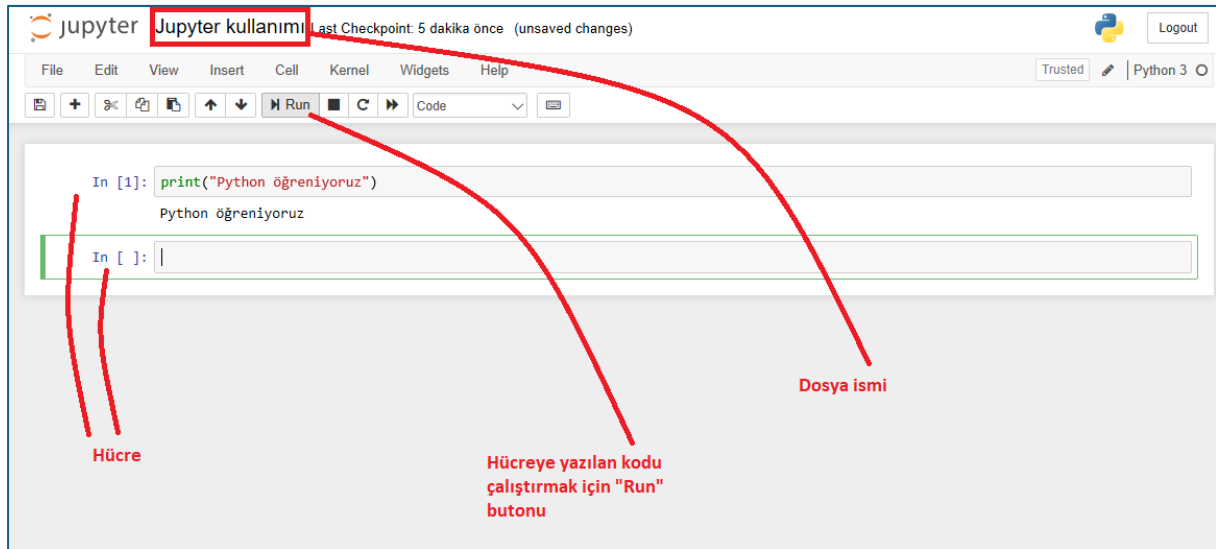
Jupyter Notebook açıldığında **C:\Users\kullanıcı ismi>** olarak açılır ve adres satırında da **http://localhost:8888/tree** olarak görünür. Eğer web arayüzü ile Jupyter notebook açılmazsa, Jupyter notebooku açmanın bir üçüncü yolu da, serverda görünen adresi **Ctrl+C** ile kopyalayıp web arayüzündeki adres satırına bu adres yazılırsa Jupyter notebook yine açılacaktır.

Jupyter Notebook açıldıktan sonra (kendi bilgisayarındaki) **C:\Users\kullanıcı ismi>Desktop>Python çalışmaları>Jupyter>** klasörüne gidiyorum. Jupyter Notebook' un bir özelliğide yapılan her işlemi otomatik kaydetmesidir. Yeni çalışma dosyası açmak için sağ üst köşeden New seçeneği ile Python 3 dosyası seçilmeli ve Untitled bölümüne giderek uygun bir isim verilmelidir.

Jupyter notebookun birçok özeliği vardır ve bunları akılda tutmak zor olabilir. Bu amaçla bir el kitabı hazırlanmıştır. Bu el kitabına, <https://jupyter.brynmawr.edu/services/public/dblank/Jupyter%20Notebook%20Users%20Manual.ipynb> adresinden ulaşılabilir. Böylece detaylı kullanım için gereken bilgiler buradan elde edilebilir.

Bölüm 2. Jupyter Notebook

Basit bir örnek aşağıda verilmiştir.

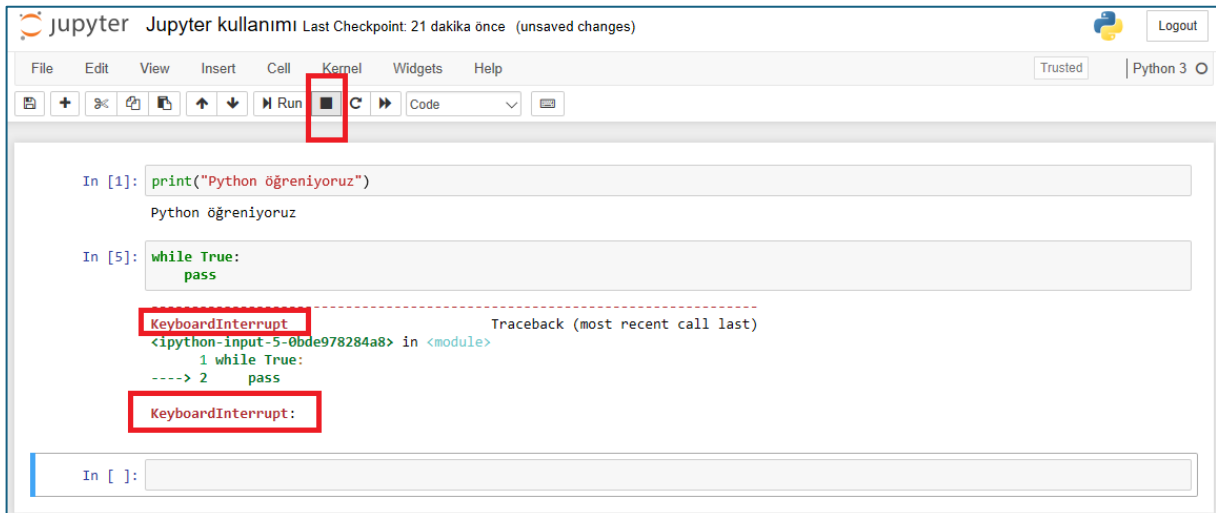


In [] yazan bölümler hücre olarak adlandırılmaktadır. Uygun formatta yazılan komutu çalıştırmak için Run Cell (Hücreyi çalıştır) anlamında şekilde gösterilen butona basılır. Bu basit uygulamada

print() komutu ekrana programın çıktılarını yazdırmak amacıyla kullanılan komuttur. Yazdırılacak bilgi tırnak içerisinde yazılır. Örneğin yukarıdaki uygulamada **Python öğreniyoruz** çıktısı programın çalıştırılmasıyla yazdırılmıştır.

Hücre çalıştırdıktan sonra yeni bir hücre oluşturulur. Dikkat edilirse, hücre çalıştırılmadan önce köşeli parantez içerisinde herhangi bir rakam yokken, hücre çalıştırdıktan sonra köşeli parantez için 1 rakamı yazılmıştır. Bu, çalıştırılmış ve işlemi yapılmış hücre sayısını göstermektedir. Run düğmesine basıldıktan sonra çalışan bir hücrede köşeli parantez içerisinde * görünür. *, hücre üzerinde hala işlem yapıldığı anlamına gelmektedir. Hücreyi çalıştırmak için Run butonu kullanılabileceği gibi klavyeden **Shift+Enter** tuşlarına da basılabilir.

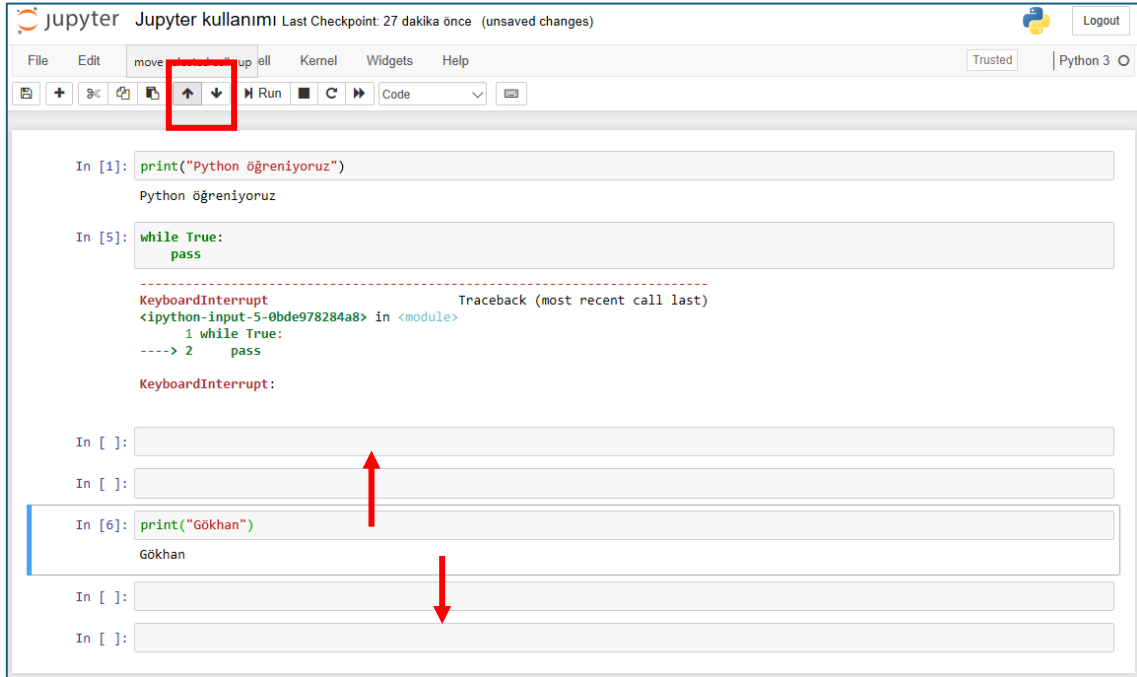
Bazı durumlarda program kısır döngüye girebilir veya programın sonuç vermesi uzun sürebilir. Bu tip durumlarda Run düğmesinin yanındaki **Interrupt** (*kesmek, yarıda kesmek*) düğmesine basılabilir.



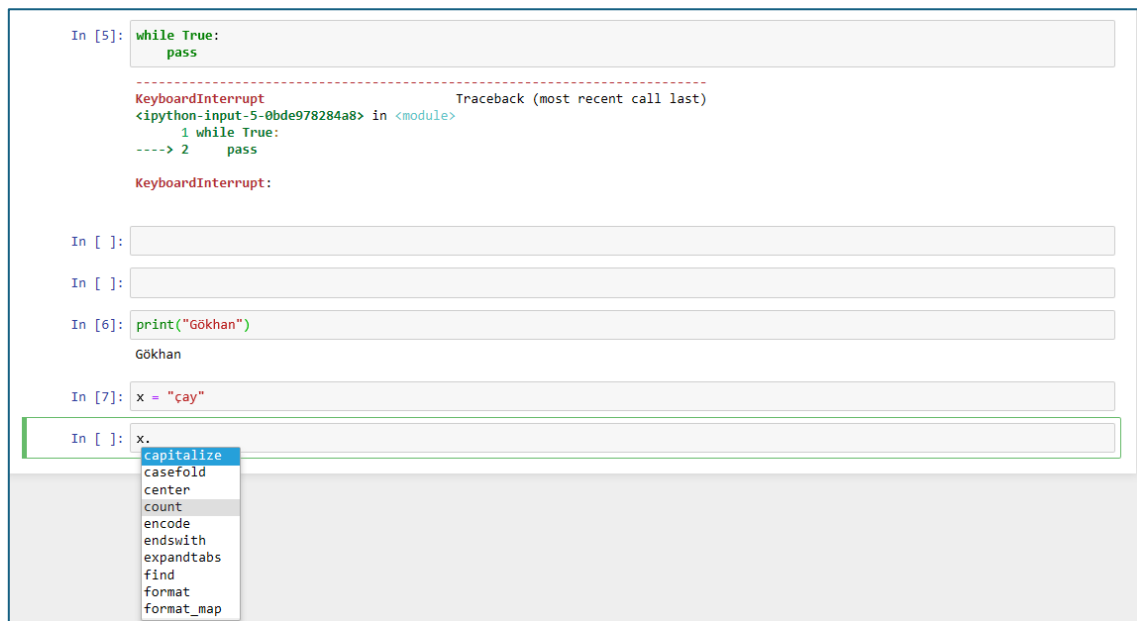
Bölüm 2. Jupyter Notebook

Interrupt seçeneği ile program durdurulduğunda ya da yarıda kesildiğinde dikkat edilirse hücrenin altında **KeyboardInterrupt** ve diğer bazı bilgiler gösterilmektedir. Aynı işlem **Kernel** sekmesinden **Interrupt** denilerek de yapılabilir.

Bir hücrenin altına sol üstteki + düğmesine basarak gerektiği kadar hücre eklenebilir. Hücrelerin yerlerini değiştirmek için de Run tuşunun sol tarafındaki yukarı ve aşağı ok düğmeleri kullanılabilir.



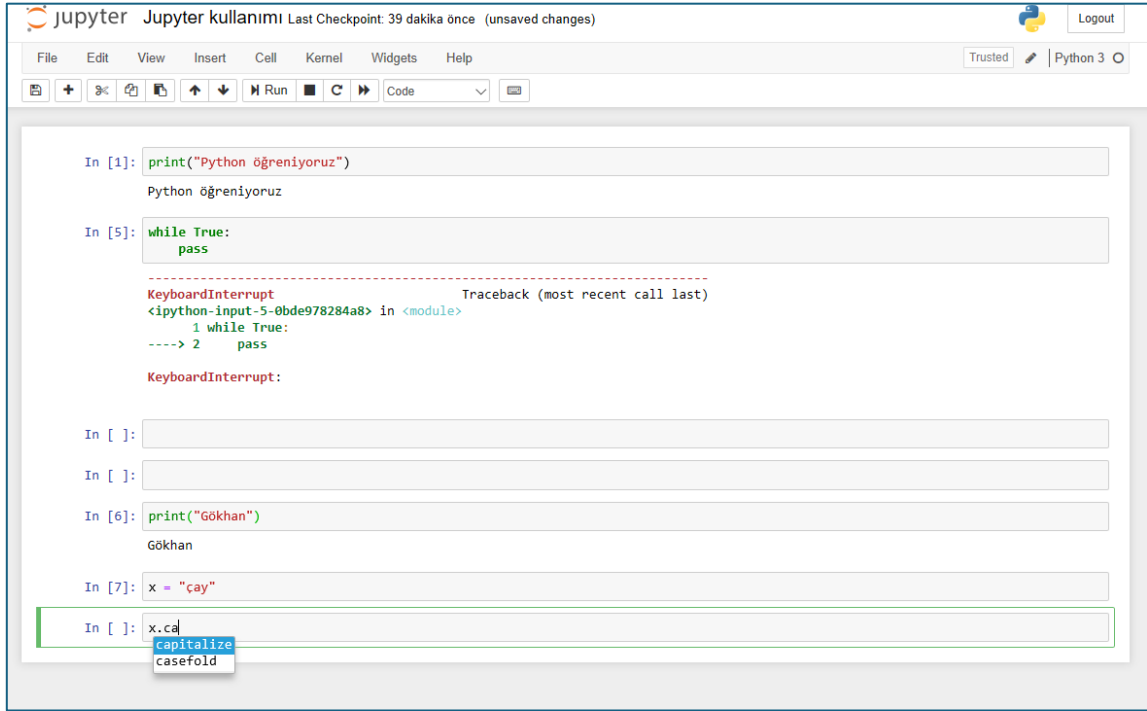
Jupyter notebookun bir diğer özelliği de aşağıdaki örnekte gösterilmiştir.



Bölüm 2. Jupyter Notebook

Bu örnekte bir **string (kelime)** değişkeni olarak **x** değişkeni tanımlanmıştır. Bu değişken ile işlem yapmak için bir sonraki hücrede **x.** yazdıktan sonra **TAB** tuşuna basılırsa **x** değişkeni üzerinde yapabileceğimiz işlemler küçük pencere açılarak görüntülenir. Değişken üzerinde nasıl bir işlem yapılmak isteniyorsa fare yardımıyla bu işlem açılan pencereden seçilebilir.

x.ca yazılıp **TAB** a basıldığında **ca** ile başlayan tüm işlem seçenekleri gösterilebilir.



```
In [1]: print("Python öğreniyoruz")
Python öğreniyoruz

In [5]: while True:
        pass

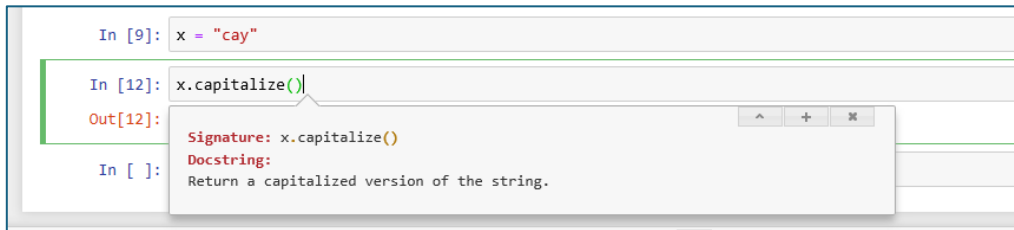
KeyboardInterrupt                                Traceback (most recent call last)
<ipython-input-5-0bde978284a8> in <module>
      1 while True:
----> 2     pass

KeyboardInterrupt:

In [ ]:
In [ ]:
In [6]: print("Gökhan")
Gökhan

In [7]: x = "çay"

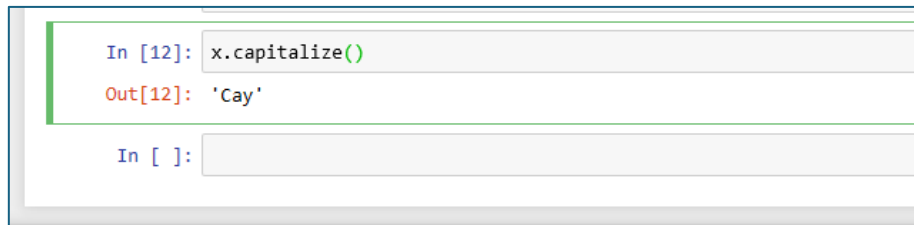
In [ ]: x.ca
        capitalize
        casefold
```



```
In [9]: x = "çay"

In [12]: x.capitalize()
Out[12]:
Signature: x.capitalize()
Docstring:
Return a capitalized version of the string.
```

Anlaşılabacağı üzere, capitalize komutu, **x** string değişkeninin ilk harfini büyük yapmıştır.



```
In [12]: x.capitalize()
Out[12]: 'Cay'

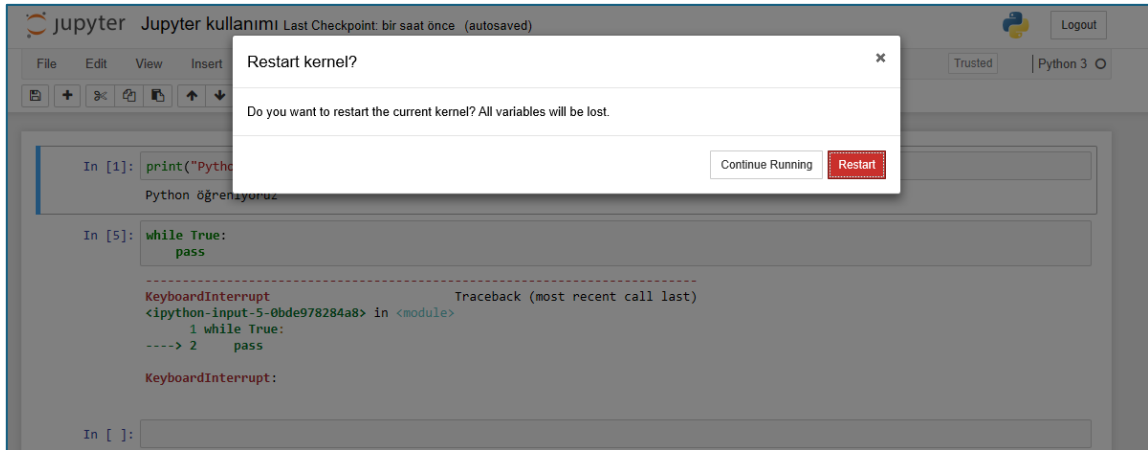
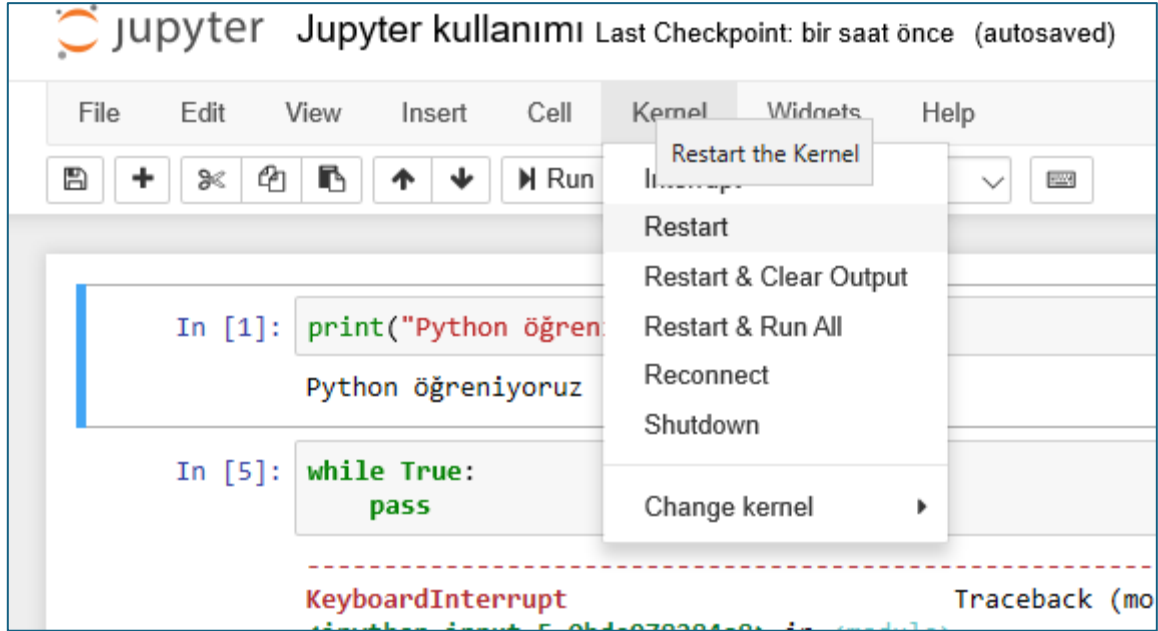
In [ ]:
```

Ayrıca, Python'da help şeklinde bir fonksiyon bulunmaktadır. Hücreye **help(x)** yazıldığında **x** olarak oluşturduğumuz **string** değişkenin bütün özellikleri dökülür. Bu işlem herhangi bir veri tipi içinde yapılabilir. **help(x.count)** denilirse **x** string değişkenimizin count metodunun özelliklerini öğrenmek istersen **Shift+TAB** yapabiliriz. Aynı işlemi, hücreye **help(x.count)**

Bölüm 2. Jupyter Notebook

yazarakta program çıktısı olarak alabiliriz. Böylece count fonksiyonun ne iş yaptığını görebiliriz.

Çalıştığımız dosyadaki komutları değil ancak diğer şeyleri temizlemek istersek bu durumda **Kernel** sekmesinde **restart** seçeneği seçilmelidir. Bu işlem yapılırsa, her kod yani her hücre yeniden çalıştırılmalıdır, çünkü tüm değişkenler kaybedilecektir.



Bölüm 2. Jupyter Notebook

Python Temel Kavramlar

Sayılar, Değişken Tanımlama, Temel Matematik Operatörler

Python’ da sayılar: Tam sayılar (integer) ve ondalıklı sayılar (float) Python’ da bir veri tipidir.

Temel Matematik operatörler: Toplama (+), Çıkarma (-), Çarpma (*) ve Bölme (/)

Değişken tanımlama: Değişken, bir veri tipinden (sayısal (numeric) ya da sözcük (string)) değer tutan birimlerdir ya da objelerdir. Sayısal bir değişken, **değişkenin ismi = sayı değeri** olarak tanımlanır.

Değişken ismi verilirken aşağıdaki durumlara dikkat edilmelidir.

- Değişken ismi sayı ile başlayamaz.
- Değişken ismi, isim tamlaması gibi ise kelimeler arasında boşluk bulunamaz.
- :",<>/?\\(!@#\$%^&*~--+ simgeleri değişken isminde kullanılamaz. Sadece alt çizgi _ kullanılabilir.
- Python’ da tanımlı **while**, **not** gibi özel kelimeler değişken ismi olarak kullanılamaz.

Aslında bu kurallar tüm bilgisayar programlarında geçerlidir.

Python’da veri tipi dinamikdir. ***Yani tamsayı değeri verilmiş bir değişken programın işleyişinde örneğin ondalıklı bir sayı olarak dinamik olarak güncellenebilir.***

Not: Bölme işlemi sonucunda Python’un özelliği olarak sonuçlar float yani ondalıklı olarak görüntülenir. Bu özellik Python 3 ile gelen ve sonuçların daha kesin gösterilmesi için konulmuş bir özelliktir.

Python’ da değişken isminde büyük küçük harf duyarlılığı bulunmaktadır. Yani x ve X farklı değişken isimleri olarak kullanılabilir.

Bir değişkenin değerini bir artırmak için (ki bu işlem döngülerde çok önemlidir) **+1** yapılabilir. Ancak Python’ da daha kolay bir yöntem **+=1** tanımıdır. Bu tanım ile değişkenin değeri bir artırılır ve yeni değişken değeri olarak atanır. Sayı iki artırılmak istenirse de **+=2** yazılabilir vb.

Bir değişkenin bir sayı ile çarpılıp çarpımın yeni değişken değeri atanması için de ***=sayı** komutu kullanılabilir. Örneğin **x=5** ise ve yeni değeri **x=x*4** ise bu, **x*=4** olarak yazılabilir.

Ya da bir x değişkenin kendi değerinden örneğin 3 sayısı çıkarılacaksa, **x-=3** komut satırı kullanılabilir.

Yorum satırları

işaretinden sonra (Alt Gr+3) tek satırlık yorum satırı eklenebilir.

tekli yorum satırı

Çoklu yorum satırları için de

“““

Çoklu yorum satırı

”””

Bölüm 2. Jupyter Notebook

özellği kullanılır. Yorum satırları Python tarafından işleme alınmaz.

Matematik Operatörleri

// sembolü tamsayı bölmesidir.

% sembolü de bölümden kalanı göstermektedir.

Üs bulma işlemi için iki tane ** işareti peş peşe kullanılır.

Köklü sayılar için de örneğin karekök için `**(0.5)` yazılarak sayının karekökü belirlenir.

Python'da veri yapıları aşağıda verildiği gibidir ve amaca uygun olarak bu veri tiplerinden biri ya da bir kaç ile çalışmak gerekir.

