FIZ220_EST_UygulamaNotlari_01_Pythona_Giris

June 16, 2020

1 Uygulama Notları: 1

1.1 FİZ220 - Bilgisayar Programlama II | 06/03/2020

Python'la 'Merhaba Dünya!'

- Python'a Giriş
- Python hakkında çok kısa bilgi
- Python'ın popülerliği
- Hangi Python IDE'sini kullanmalıyız?
- Jupyter ortamı
- Hücreler
- Python kütüphaneleri
- Jupyter sayfanızı aktarma

Dr. Emre S. Taşcı, emre.tasci@hacettepe.edu.tr
Fizik Mühendisliği Bölümü
Hacettepe Üniversitesi

2 Python'a Giriş

2.1 Python hakkında çok kısa bilgi

Python dili 1990 yılında, Guido van Rossum tarafından geliştirilmiştir. Yüksek seviye bir dil olup, işlemci tarafında pek çok zahmetli işlem (örn. *hafıza ayarlamaları, çöp toplama*) sistem tarafından otomatik olarak halledilmektedir. Ayrıca nesne tabanlı bir yapıya sahiptir (daha da ileri gidersek: Python'da hemen hemen **her şey** bir nesnedir).

Dil, ismini İngiliz anarşist komedi topluluğu Monty Python'dan almaktadır (hatta, bizzat van Rossum'un örneklerde kullanılacak değişken ve durumların da grubun skeçlerinden seçilmesi konusunda ricası vardır).

2.2 Python'ın popülerliği

Yaygın kütüphane desteği ve internetin doğrudan işletim sistemlerinde kaynak olarak kullanımının önünün açılması sonucu 2000'li yıllardan itibaren "en çok kullanılan diller" listelerinde hep üst sıraya yerleşmiştir. Bu listelerde kriter sayılan TIOBE endeksinde Python, Java (%17.8) ve C'den

(%16.3) sonra %10.1'lik kullanımla 3. sırada yer almaktadır ve yükselişini kararlı bir biçimde sürdürmektedir. GitHub ve Stack Overflow istatistiklerinden yola çıkılarak hazırlanan bir diğer endeks olan RedMonk sıralamasında ise Ocak 2020 itibarı ile JavaScript ile birinciliğe başabaş oynamaktadır. İnternette en çok başlangıç ve tanıtımları ziyaret edilen diller üzerinden sıralama yapan PYPL endeksinde ise açık ara ile (Python: %30.1; Java: %18.8) birinci sıradadır. *(Bütün istatistikler 2020 Mart ayı açıklamalarına dairdir)*

2.3 Hangi Python IDE'sini kullanmalıyız?

Temel olarak Python yorumlanarak çalışan ("interpreted") bir dil olduğundan, bir yorumcuya ("interpreter") ihtiyaç duyar. Python'ın kendisi ile gelen *IDLE* IDE'si temelde iş görse de, yaşamı kolaylaştırıcı pek çok özellikten mahrumdur. Bu nedenle, zorunlu durumlar dışında pek kullanılmaz.

Python'da program geliştirmek için yoğun olarak PyCharm ve IPython IDE'lerini kullanmaktadırlar. Anaconda ise, IPython IDE'sini (Jupyter üzerinden) ve pek çok popüler sayısal analiz kütüphanesini içinde barındıran bir yazılım paketi olup, kullanım kolaylığı açısından benim de tavsiye ettiğim IDE'dir.

Jupyter yoluyla bir web tarayıcısında yazılan Jupyter defterleri (*ipynb - IPython Notebooks*) gerek görsel, gerekse kullanım açısından gerçekten de akıcı bir geliştirmeye olanak tanır (bu uygulama notları da bizzat Jupyter ile hazırlanmaktadır 8).

3 Jupyter ortamı

3.1 Hücreler

Jupyter'ı bir kelime işlemci (örn: MS Word) belgesi olarak düşünebilirsiniz: bu dökümanın bazı yerlerine açıklamalar yazıp, aralara da resimler eklediğiniz gibi, bir Jupyter defterinin istediğiniz kısımlarına yazı yazıp, istediğiniz kısımlarında da programınızı yazıp çalıştırabilirsiniz – tıpkı şu anda yapmakta olduğum gibi. Python'da basit bir işlem yapalım:

[1]:
```
5 + 7
```

[1]:
```
12
```

Yukarıdaki işlemin olduğu kısım, şu anda bu satırların olduğu kısım gibi, çalışmamızın bir parçası olup, "hücre" (*cell*) adıyla anılırlar. Bir hücre şu üç çeşitten biri olabilir: 1. **Kod Hücresi:** Bu hücre -doğal olarak- en temel hücre tipimiz olacaktır. Buraya yazılan kodlar çalıştırılıp, çıktıları da aynı hücrenin altında monte şekilde gösterilecektir. (Kısayol tuşu: **Y**) 2. **Metin Hücresi:** Metin hücreleri de şu anda okumakta olduğunuz hücreler gibi, bilgi paylaşımı amacıyla oluşturulan hücrelerdir. Bu kısımları programlarda açıklamaları yazdığımız yorum satırlarının gelişmiş versiyonları olarak düşünebiliriz. **Kalın**, *italik* vs. şeklindeki basit biçimlendirmelere de izin verirler (bu biçimlendirmeler için kullanılan notasyona **MarkDown** notasyonu adı verilmekte olup, detaylı açıklamalar ve örnekler için Adam Pritchard'ın kapsamlı sayfasına bakabilirsiniz. Metin hücreleri, MathJax motoru sayesinde, karmaşık formülleri LaTeX yoluyla yazmanıza da izin vermektedir, örneğin: $\oint E \cdot \mathrm{d}A = \frac{\rho}{\epsilon_0}$. (Kısayol tuşu: **M**) 3. **Çiğ Hücre:** Çiğ hücreler bir kodu (veya

şiiri ;) şeklini bozmadan, normalde çeşitli anlamlara gelen sembolleri olduğu gibi girmenize yarar. Şimdilik pek kullanacağımız bir hücre tipi değil. (Kısayol tuşu: **R**)

Hücrelerle ilgili çok kullanacağınız birkaç diğer kısayol tuşu ise şunlardır: * <Enter> : Seçili hücreyi değiştirme moduna alır. * <CTRL>+<Enter> : Seçili olan hücreyi *çalıştırır* (kod hücresi ise yazılı kodu çalıştırıp, çıktısını döndürür; metin hücresi ise markdown ve LateX sembollerini işler) * b (below/aşağı) ve a (above/yukarı) : Seçili olan hücrenin aşağısına (veya yukarısına) yeni bir hücre ekler. * <CTRL>+<Shift>+<s> (split) : Seçili olan hücreyi o anda imleçin olduğu satırdan iki ayrı hücreye böler. * h (help) : Kısayol tuşlarının karşılıklarını gösterir.

Bütün defteri çalıştırmak için ise **Kernel** menüsünden "Restart & Run All" diyebilirsiniz - bu durumda, defteri yeni açmışsınız gibi çekirdek (*kernel*) baştan yüklenir ve her şey sıfırdan çalıştırılır; **Cell** menüsünden "Run All" seçeneği ise, hücreleri en tepeden aşağıya doğru sırayla çalıştırır fakat daha önce çalıştırdığınızda edinilen (değişken değerleri gibi) bilgileri de kullanır. Kod hücrelerinin sol başlarındaki sayılar hücrelerin hangi sırayla çalıştırıldıklarını işaret eder (beklemediğiniz bir çıktı almanız halinde gidişatı kolayca takip edip, durumu anlayabilmeniz için).

4 Python kütüphaneleri

Python, çeşitli işlere özgü (örn: matris hesabı, grafik çizimi, veritabanı, konumsal işlemler, görüntüleme, ses işleme, yapay zeka vs.) metotları barındıran pek çok kütüphane ile desteklenir; zaten bu kadar geniş kullanıcı sayısına ulaşmasında da bu çeşitliliğin büyük katkısı vardır. Python'da kütüphaneler *modül* (*module*) olarak adlandırılır.

Sayısal hesap işlemlerinde yoğun olarak kullanacağımız üç adet kütüphane: 1. NumPy : Temel matematiksel fonksiyonları ve gelişmiş dizi & matris tiplerini, işlemlerini içerir. GNU Octave'da yaptığımız hemen her şeyi bu kütüphanedeki tanımlar çerçevesinde yapabiliriz. 2. SciPy : NumPy'ın bir üst kütüphanesi olarak düşünebiliriz. İleri fonksiyonlar ve metotlar bu kütüphanede bulunur, hayli kapsamlıdır. 3. MatPlotLib : Grafik çizmek için GNU Octave / MATLAB benzeri komutlarla işimizi kolaylaştırır.

Python'da istediğimiz kütüphaneyi:
`import <kütüphane-adı> as <kütüphane-adı-kısaltması>`
veya
`from <kütüphane-adı> import <istenilen-metotlar>`
biçimiyle çağırabiliriz. İki kullanım biçimi birbirinden farklıdır: İlkinde kütüphane elemanlarını çağırırken kütüphanenin kısa adıyla işaret ederiz (bu tür aidiyete "isimuzayı" (*namespace*) denir); ikinci türde ise kütüphane elemanları doğrudan ana isimuzayına eklenir, ilgili kütüphaneyi işaret etmemiz gerekmez. İkinci yaklaşım daha pratik gelse de birden fazla kütüphanenin aktarıldığı durumlarda karışıklığa yol açabilir, bu nedenle mümkün mertebe ilk yaklaşımı kullanmak gerekir.

Not: ikinci yaklaşımla sadece bir tek metot eklense bile, Python yine de bütün kütüphaneyi yükleyecektir. `import numpy as np` ile `from numpy import pi` arasında performans bakımından hiçbir fark yoktur: iki durumda da bütün NumPy modülü yüklenir - ikinci durumda sadece *pi*'nin kullanımı mümkün olsa bile!

[2]:
```
# Örnek:
print(pi) # tanımlı olmadığından hata gelecektir
```

```
␣
↪---------------------------------------------------------------------------

       NameError                                 Traceback (most recent call␣
↪last)

       <ipython-input-2-71aa5d951ad9> in <module>
         1 # Örnek:
    ----> 2 print(pi) # tanımlı olmadığından hata gelecektir


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

[3]:
```
# Kütüphaneyi 'np' kısaltmasıyla aktaralım:
import numpy as np
print(np.pi) # pi de tanımlı,
print(np.e) # e de...
```

```
3.141592653589793
2.718281828459045
```

[4]:
```
# Kütüphaneden sadece pi'yi aktarıp, öyle çağıralım:
from numpy import pi
print(pi) # pi tanımlı,
print(e) # fakat e değil...
```

```
3.141592653589793
```

```
␣
↪---------------------------------------------------------------------------

       NameError                                 Traceback (most recent call␣
↪last)

       <ipython-input-4-ec3cedfb23f3> in <module>
         2 from numpy import pi
         3 print(pi) # pi tanımlı,
    ----> 4 print(e) # fakat e değil…


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

5 Jupyter sayfanızı aktarma

Jupyter'da hazırladığınız sayfalar otomatik olarak sıklıkla kaydedilir (bunun yanısıra, siz de sol üstteki ikondan veya <CTRL>+s kestirmesinden dilediğiniz zaman kaydedebilirsiniz).

Dosyanızı bir başkasına doğrudan veya farklı bir biçimde göndermek içinse **File** menüsünden "Download as…" seçeneğini kullanabilirsiniz. Bu menüde çıkan seçeneklerden başlıcaları:

- Notebook (.ipynb) : Oluşturmuş olduğunuz defterin tam olarak bir kopyasını verir.
- Python (.py) : İlgili kod kısımlarını tutar, yazı kısımlarını ise yorum olarak işleyip verir.
- HTML (.html) : Grafikler de dahil olmak üzere, bütün içeriği web tarayıcınızda gördüğünüz şekilde *statik olarak* kaydeder fakat kod doğrudan çalıştırılamaz ya da siz defterdekine benzer şekilde işlemler yapamazsınız.
- PDF (.pdf)