

Pyhton Programlama

Berkay Dedeođlu

Haziran, 2014

İçindekiler

1	Giriş	2
1.1	Nasıl Python Öğreneceğiz?	2
1.2	Bu Kaynaktan En Verimli Şekilde Yararlanmak İçin	2
1.3	Programlama	2
1.4	Programlama Nasıl Yapılır?	3
1.5	Çok Bilinen Programlama Dilleri	3
1.6	Makine Dili Nedir?	3
1.7	Python	4
1.8	Neden Python?	4
1.9	Başka Hangi Kaynaklardan Faydalanabilirim?	4
1.10	Bilinen Python Programları	4
2	Python	6
2.1	Python'a Giriş	6
2.2	Gözden Geçirme	6
2.3	Python 2 ve Python 3 Farkları	6
2.4	Kurulum	7
2.5	Uçbirim ve Etkileşimli Kabuk	8
2.6	Python ile Programlama Mantığı	10
2.7	Programlama yaparken karşılaşılan hatalar	12
2.8	Âdettendir... 'Merhaba Dünya'(Hello World)	13
2.9	Print(Yazdır) Fonksiyonu:	13
2.10	Input (Giriş) Fonksiyonu	16
2.11	Hesap Makinesi Python!	17
2.12	Karakter Dizilerinin Toplama ve Çarpma Özelliği:	18
3	Kapanış	20

1 Giriş

Ubuntu-tr gibi harika bir platformun içine girdikten sonra insan, bir şeylere katkı yapmadan, yardım etmeden durmıyor. Velhasıl ben de bu katkı payımın küçük bir kısmını burada kullanmaya karar verdim. Programlama, Python ve kodlar...

Python öğrenmeye geçmeden önce dilerseniz bu seri boyunca öğrenimin amacı ve nasıl işleyeceği hakkında biraz bilgi sahibi olalım:

1.1 Nasıl Python Öğreneceğiz?

Python esasında çok basit, eğlenceli ve çabuk bir programlama dilidir. Bu anlatım, Python programlama dilini hiç bilmeyen kullanıcıların dahi anlayabileceği bir seviyede sürecektir. Anlaşılmayan kısımların Ubuntu-tr forum sayfasına göz atarak rahatça çözülebileceğini düşünüyorum.

Bu anlatımda bulunan her şeyi ayrıntılı ve görsellerle desteklenmiş biçimde bulacaksınız. Anlatılan konular, hiyerarşik biçimde alt konular ile desteklenmiş olacaktır. Her sayının benim için ayrılmış kısmının sonunda o güne kadar işlenmiş konuları kapsayan birkaç uygulama soruları olacaktır.

Bunların cevapları birden fazla olmakla beraber, bir programcı için en ideal olanı bir sonraki sayıda verilecektir. Elbette ki programınız istenilenleri hatasız biçimde karşılıyorsa, cevabınız tam anlamıyla doğru demektir. Ayrıca bu anlatım hakkında sorunlarımızı, önerilerinizi forumumuzun 'SUDO E-Dergi' başlığı altında bildirebilirsiniz. Bu tür geribildirimlerin bizi çok memnun edeceğini belirtmeden geçemem.

1.2 Bu Kaynaktan En Verimli Şekilde Yararlanmak İçin

Eğer gerçekten iyi programlar yapmak ve kaliteli bir Python programcısı olmak istiyorsanız bu kaynaktan en verimli şekilde yararlanmanız gerekir. İşte tam da bunun için bu kaynağın yazarından öneriler:

- Anlatım kısımları tam olarak anlaşılmalıdır.
- Bu anlatım, bilgisayara erişiminiz varken okunmalıdır. Bu sayede öğrendiklerinizi hemen deneyebilirsiniz.
- Anlatımdaki örnekler tam olarak anlaşılıp benzerleri oluşturulmalıdır.
- En sondaki uygulama soruları üzerinde çalışılmalıdır.

1.3 Programlama

İnsanlar en hızlı ve akıcı olarak dil ile anlaşılır. Dil olmazsa insanlar birbirlerini anlayamaz ya da yanlış anlar. Bununla birlikte insanlar coğrafi, kültürel, dinsel vb. etkenlere göre çevrelerindeki insanlarla anlaşabileceği diller oluşturmuşlardır. Örneğin Araplar coğrafi koşullar nedeni ile develeri hayatları içinde oldukça fazla kullanırlar. Develerin bu kadar çok olduğu bir toplulukta develer dilde büyük yer edinmelidir. Aksi takdirde bölge insanları iletişimi zor ve uzun olur. Bu yüzden ki Arapçada deve, 200 küsur farklı isimle anılır. Bu bir ihtiyaçtır.

Tıpkı bu şekilde insanların bilgisayar denilen alete istediklerini yaptırılmaları için diller oluşturulmuştur. Bunlar o kadar çoktur ki bazılarını yalnızca bir iki kişi bilirken, bazılarını ise dünya üzerindeki programcılar âdeta adları gibi bilir. Her bilgisayar dilinin temelini İngilizce oluşturur. Ayrıca en bol kaynak yine İngilizce dilindedir. Arapça örneğinde verdiğimiz gibi bu bilgisayar dilleri bazı etkenlere göre gelişmiştir. Örneğin Python dahil birçok programlama dilinde oyun yapmak mümkün, hatta oldukça kolay olsa da bu iş için C/C++ dili en çok kullanılır. Ya da her programlama dilinde bilimsel işlemler yapılsa da Matlab bu iş için en çok kullanılanlardandır.

1.4 Programlama Nasıl Yapılır?

Aslında bakarsanız programlama bilgisayara uygun dilde verilen emirlerle yapılır. Programlama esnasında bu emirlerin çoğunu programcının vereceği gibi bazılarının da programı kullanan kişi tarafından verilmesi beklenir. Programlama çok kolay olsa da herkesin programcı ya da bilgisayar kurtları olmasını bekleyemezsiniz. Kullanıcının vermek istediği emri kodlara dönüştürecek siber yapılara program ya da yazılım denir. Bu programlar programcılar tarafından belirli kodlar yazılarak oluşturulur. Zaten programlama dili öğrenmek demek esasında bu kodları öğrenmek demektir. Daha sonra deneyim kazandıkça programlama mantığı öğreniliyor.

Sonuç olarak programlama aslında bilgisayara verilen emirlerdir. Bunu bilgisayarın anlayacağı dilde vermek gerekir. Örneğin Gedit'i açıp;

```
"Ekрана 'Merhaba Dünya' yaz"
```

demek ile, sadece Japonca bilen birine Türkçe 'İğneyle kuyu kazmak' deyimini anlatmak aynı şeydir. Eğer bunu Python ile yapmak istersek ileride öğreneceğimiz biçimlerde;

```
1 print ("Merhaba Dünya")
```

yazmalıyız. Tabii bilgisayara emir vermek sadece Python ile değil birçok programlama dili ile mümkün. Dilerseniz bu programlama dillerinin bazılarını ufaktan değinelim.

1.5 Çok Bilinen Programlama Dilleri

- **C/C++:** Birçok programlama dilinin atasıdır. İşletim sistemleri, oyunlar genelde bu dille oluşturulur. Gücü ve hızı ile neredeyse tüm dilleri geride bıraksa da Python'a göre çok daha fazla kod gerektirir.
- **Java:** Esasında yorumlamalı bir dildir. Köklü ve sağlam bir dildir. Görsel programlama için tercih edilir. Sözdizimi C/C++ diline benzer. Güçlü bir dil olmasına karşın yine de fazla kod yazımı gerektirir.
- **Python:** Zaten bu programlama dili hakkında sayfalarca bilgi vereceğim.
- **C#/Mono:** C#, Microsoft firması tarafından geliştirilen, kolay yazımı olan güçlü bir programlama dilidir. C# kodları Windows platformu dışında tam olarak çalışmaz. İşte C# uygulamalarını Linux altında çalıştırmak için Mono projesi geliştiriliyor. Ancak hâlen tam yeterli seviyeye ulaşamamıştır.
- **Pascal:** Öğrenimi oldukça basit olan bir programlama dilidir. Mac işletim sistemlerinde hâlâ etkin düzeyde kullanılır.
- **Assembly:** Makine diline en yakın dil bu dildir. Makine dilinden sonra en çok kod bu dilde yazılır. Programlama oldukça yorucudur. Mecbur kalınmadıkça tercih edilmez.

1.6 Makine Dili Nedir?

Makine dili bilgisayarın anlayabildiği tek dildir. Yukarıda bahsettiğim diller çeşitli yollarla makine diline çevrilmeden bilgisayar bunu anlayamaz.

Mutlaka duymuşsunuzdur; "Bilgisayarlar sadece sayılarla çalışır: 0 ve 1". Bu söylem gittikçe popülerleşse de aynı hızda doğruluğu da azalmaktadır. Şu anda bazı markalar ve bir iki programcı dışında makine dili kullanılmıyor. Öyle ki makine dili ile yazılmış bir program satıyorsanız hiç şüphelenmeden alıcıya: "Bu programı kodlayan çocuk kör oldu!" diye reklamını yapabilirsiniz.

Sonuç olarak makine dili ile kodlamanın ve okumanın zor olması, programın her platformda çalışabilmesi gibi sebeplerden ötürü programlama dilleri geliştirilmiştir.

Programlama hakkında çok konuştuk. Artık asıl konumuza ilk adımı atalım.

1.7 Python

Python programlama dili güçlü, kullanışlı, çabuk yazılabilen, eğlenceli ve kolay okunabilen, yorumlamalı bir programlama dilidir. Python kodları Python yorumlayıcısıyla C diline çevrilip çalıştırılır.

Python projesi 1990 yılında Guido Van Rossum adında Hollandalı genç tarafından başlatılmıştır. Adı her yerde de söylendiği üzere piton yılanından gelmez. Adını Monty Python adlı komedi ekibinden alan Python programlama dili şu anda elemanlarının gönüllülerden oluştuğu 'Python Yazılım Vakfı' tarafından geliştirilmektedir.

1.8 Neden Python?

Yukarıda da bahsettiğim gibi Python kolay ve bir o kadar da güçlü bir programlama dilidir. Geniş bir kullanıcı kitlesi vardır. Hemen her işletim sisteminde çalışır. Hâlâ geliştirilmeye devam etmesi ve büyük otoriteler tarafından bu projeye destek verilmesi Python'u bu seviyeye getirmiştir.

İlerki alıştırmalarımızda zaten hız testi yaptığımızda göreceksiniz ki Python, diğer yorumlamalı dillerin birçoğundan çok daha hızlıdır. Ayrıca Python son moda ayak uydurup %100 nesne tabanlı olarak geliştirilmiştir.

1.9 Başka Hangi Kaynaklardan Faydalanabilirim?

Python'un Türkçe kaynağı oldukça azdır. Azdır ama diğer programlama dillerine göre daha düzenlidir. Python öğrenmek için buradan başka baştan sona anlatabilen iki Türkçe kaynak var:

- [İstihza](#)
- [Mustafa Başer Python](#)

Bunlar dışında Google'da ufak bir arama ile pek çok faydalı ancak çoğu yarım kalmış kaynak bulabilirsiniz. Bunlar dışında Python'u daha çok belgeyle ele alan orijinal Python sitesini ziyaret edebilirsiniz. Bunun gibi internet üzerinde İngilizce çok kaynak var, biraz İngilizce bilgisiyle bu kaynaklar ile haşır neşir olabilirsiniz ne mutlu size.

Şimdi diğer konuya geçmeden önce Mustafa Başer ve Fırat Özgül'e katkılarından dolayı teşekkürü borç bilirim. Umarım onların gibi özgün ve anlaşılır bir kaynak oluşturabilirim.

1.10 Bilinen Python Programları

Python ile teorik olarak birçok şey yapılabilir. Ancak daha çok kısa ve çabucak bitirilmesi gereken işler için seçilir. Bununla birlikte Python ile yapılmış oldukça büyük projeler de vardır:

1. İstihza.com
2. Instagram
3. Metin2 oyunun büyük bölümü
4. Ubuntu Yazılım Merkezi
5. Ubuntu Kurulum Sihirbazı
6. GIMP'in birçok betiği
7. Blender'in büyük bir kısmı
8. Birçok GNU/Linux dağıtımının büyük bir kısmı
9. Google ve Youtube'un da Python kodlarını büyük oranda kullandığı söyleniyor. Buna Guido Van Rossum'un bir zamanlar Google'da çalışması önyak oluyor.

Bunlarla birlikte aslında sayfalarca proje listelenebilir ama asıl amacımız bu olmadığından bu kadarla yetinelim. Ancak bilmeniz gereken bu örneklerin bazıları hakkında resmî açıklama gelmemiş olmasıdır. Buna Google örneği verilebilir.

Python hakkında bu kadar ön bilgi yeterli. Eğer bunu okuyan kişi bu bilgiler ile Python'u merak etmiş ve hatta heyecanlandıysa haydi hemen programlama öğrenelim. Sabırsızlanıyorum... Haydi Başlayalım!!

2 Python

Artık Python programlama diline başlama vakti geldi. Programlama serüvenimiz başlasın....!!!

2.1 Python'a Giriş

Bu kısım Python programlama diline giriş niteliğinde olacaktır. Serinin bu sayısında Python programlama diline sadece giriş yapacağız. Programlama öğrenirken olmazsa olmaz 'Merhaba Dünya' uygulamamızı ve kullanıcıdan veri alma işlemlerini öğreneceğiz.

Genellikle son kullanıcı kitlesi bu kısımda bir hayal kırıklığına uğruyor. Çünkü burada yaptıklarımız bir pencerede değil yalnızca uçbirim ekranında sergileniyor. Bu durum alışık olunan bir şey değil tabii ama programlamanın ve programlama dilinin mantığını öğrenmek grafik arayüzlü programlama tasarlamaktan çok daha önce gelir. Bu kısımlar öğrendikten sonra grafik kullanıcı arayüzlü programlar çok daha basit gelecektir. Zaten bu seri boyunca en az bir grafik arayüz kütüphanesi kullanmayı tüm detayları ile anlatacağım.

Siz şimdilik burada olanlara mantık erdirmeye çalışın. Biraz daha bilgilendikten sonra grafik arayüzünün gereksiz olduğunu bile düşünebilirsiniz.

2.2 Gözden Geçirme

Python programlamadan önce son kez bu kısma kadar öğrendiklerimizi birkaç yeni bilgi ile tekrarlayalım:

- Programlama temel olarak bilgisayara verilen emirler topluluğunun yazım sürecidir.
- Bilgisayara verilen emirler ancak bilgisayarın anlayacağı dilde olursa karşılık alınır.
- Bilgisayara emir vermenin asıl yolu makine dilidir. Ancak bu dil insanlar için çok yorucudur. Bu sebeple programlama dilleri geliştirilmiştir.
- Çok sayıda programlama dili vardır. Bunların tamamı çeşitli yollar ile makine diline çevrilir.
- Kullanacağımız dil olan Python yorumlamalı bir dildir. Python kullanabilmek için bilgisayarda Python yorumlayıcısı bulunmalıdır.

Evet... Şimdi bu bilgiler ışığında yolumuza devam edelim.

2.3 Python 2 ve Python 3 Farkları

Kullanacağımız dil, Python, iki ana sürümle ayrılmaktadır: 2. sürüm ailesi ve 3. sürüm ailesi. Esasında aralarında kod yazımı olarak çok büyük farklar olmasa da normal bir Python2 programını Python3 ile yazmak pek mümkün olmuyor. Bu sürümler arası tüm farklılıklara [Python Resmi Sitesinden](#) bakabilirsiniz. Ben burada bu farklılıklardan en çok göze çarpan ve işe yarayanları dile getireceğim:

- UTF-8 kodlama diline geçildi: Python2 dilinde varsayılan kodlama dili ASCII iken Python3 ailesinde varsayılan kodlama dili UTF-8 oldu. Bu durum Türkçe programlama yapanlar için yararlı oldu. Çünkü ASCII kodlama dili Türkçe karakterleri tam olarak desteklemiyordu.
- Print bir fonksiyon oldu: Python 2 ailesinde ekranda bir çıktı göstermek için şöyle bir yol izliyorduk:

```
1 print "Merhaba Dünya"
```

Ancak Python3 ailesinde durum değişti. Print deyimi yalnızca fonksiyon oldu ve yeni, kullanışlı parametreler eklendi.

```
1 print ("Merhaba Dünya")
```

Bu durumun iyi ve kötülüğü tartışılabilir olsa da bazı parametrelerin eklenmiş olması bir avantaj.

- Modül isimleri yenilendi bu durum bizim işimize şimdilik yaramasa da ileriki konularda kodlarımıza güzellik katacak.
- Python resmî sitesinde Python 3.0 sürümünün Python 2.5 sürümünden %10 daha yavaş olduğu belirtiliyor. Tabii bu yazı Python 3.0 için. Bu yazıda aynı zamanda bu durum için çalıştıklarını ve daha iyi duruma geleceklelerini belirtmişler.

Benim bu yazıda Python3 ailesini baz alacağımı bildirerek bir sonraki konu olan Kurulum konusuna geçiyorum.

2.4 Kurulum

Python'un 2 ve 3. sürümleri zaten birçok GNU/Linux dağıtımında hazır olarak geliyor. Eğer Ubuntu ya da türevlerinden bir dağıtım kullanıyorsanız uçbirimi açıp:

```
1 ~$python
```

komutunu verdikten sonra '>>>' işaretini görebilirsiniz bu Python'un hazır olduğu anlamına gelir. Peki biz yukarıda iki Python ailesinden bahsettik, 'Bu kullandığımız hangisi?' diye sorabilirsiniz. O hâlde uçbirimde Python hazırken ('>>>' işareti) şu kodu yazıp çalıştıralım:

```
1 print ("Merhaba Dünya")
```

Eğer buraya kadar sorunsuz geldiysek şöyle bir ekran karşımıza çıkmalı:

```
berkay@berkay-A510:~$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Merhaba Dünya')
Merhaba Dünya
>>> █
```

Şekil 1: Python2Konsol

Yukarıdaki yazıya göre göre bu biraz garip kaçtı. Eğer dikkat ettiyseniz uçbirime 'python' komutu verdiğimizde bize döndürülen ilk bilgi çalışan programın Python 2.7 olduğu bilgisiydi. Biz Python3 ailesine göre print komutu girdiğimiz hâlde bize doğru çıktı ulaştı. Birde normal şekilde print deyimine bakalım:

```
1 print "Merhaba Dünya"
```

Bu kodları da yazdığımızda bize aynı çıktının ulaştırıldığını görüyoruz. Bu da demek oluyor ki Python2 ailesi her iki şekilde print kullanımını destekliyor. Şimdi asıl kullanacağımız olan Python3 sürümünü çalıştıralım. Kendimize yeni bir uçbirim penceresi açarak:

```
1 ~$python3
```


kodunu girelim, öncekine benzer bir ekran gördük. Ancak bu kez döndürülen ilk bilgi, programın Python3 ailesinden bir sürüme ait olduğunu belirtiyor. Dilerseniz aynı kodları tekrar deneyelim:

```
1 print ("Merhaba Dünya")
```

```
1 print "Merhaba Dünya"
```

Eğer kodları sırasıyla yazdıysanız en sonda şöyle bir çıktı almış olmalısınız:

```
1
2 File "<stdin>", line 1
3 print 'Merhaba Dünya'
4     ^
5 SyntaxError: invalid syntax]
```

Tebrikler! Python'da ilk hatanızı aldınız.

Eğer uçbirimde 'python' ya da 'python3' komutlarını verdiğinizde '>>>' işaretini alamayıp yerine bir hata ile karşılaşıyorsanız, büyük ihtimalle sisteminizde Python yüklü değildir. Deponuzdan 'python-3' diye aratıp indirme işlemlerini yapabilirsiniz.

Eğer Windows ya da Mac işletim sistemlerinden birini kullanıyorsanız [buradan](#) sisteminize uygun kurulum dosyasını yükleyip kolayca kurulum yapabilirsiniz.

Python'u GNU/Linux üzerinde kaynak koddan derlemek biraz daha tehlikeli ve zor olduğundan burada anlatılmayacaktır. Bununla ilgili bilgiye Ubuntu-tr Wiki sayfamızın Python maddesinden ulaşabilirsiniz.

Bu konuyu bitirmeden önce belirtmem gereken bir önemli konu ise bazı sistemlerde Python sürümlerinin farklı adlarla çağrılmasıdır. Örneğin 'python' komutu ile Python3 'python2' komutu ile Python2 çağrılabilir. Bu yüzden programı uçbirimden çağırduğunuzda döndürülen bilgiye iyice dikkat etmeniz gerekir.

2.5 Uçbirim ve Etkileşimli Kabuk

Aslında bir önceki bölümde uçbirim üzerinden etkileşimli kabuğa düştük. Düştük ama farkında değildik. Etkileşimli kabuk, Python programlama diline ve diğer bazı programlama dillerine özel çok büyük bir araçtır. Etkileşimli kabuğa kodlarımızı yazabiliriz. Yazdıktan sonra enter tuşuna basar basmaz yazılan kod işletilir. Uçbirime 'python3' (ya da sisteminize göre 'python') komutunu verdiğinizde Python3 ailesinin etkileşimli kabuğuna düşmüş oluyorsunuz. Buraya dilediğiniz kodları yazıp çalıştırabilirsiniz. Bu etkileşimli kabuk bir zaman sonra sıkıcı hâle gelecek ve yazılan kodlar anlaşılmaz hâle bürünecektir. Tabii kullanıcıya göre. Bu yüzden etkileşimli kabuğun daha cezbedici versiyonu diyebileceğimiz 'ipython' yazılımını daha en başından sunmak gerekir diye düşünüyorum.

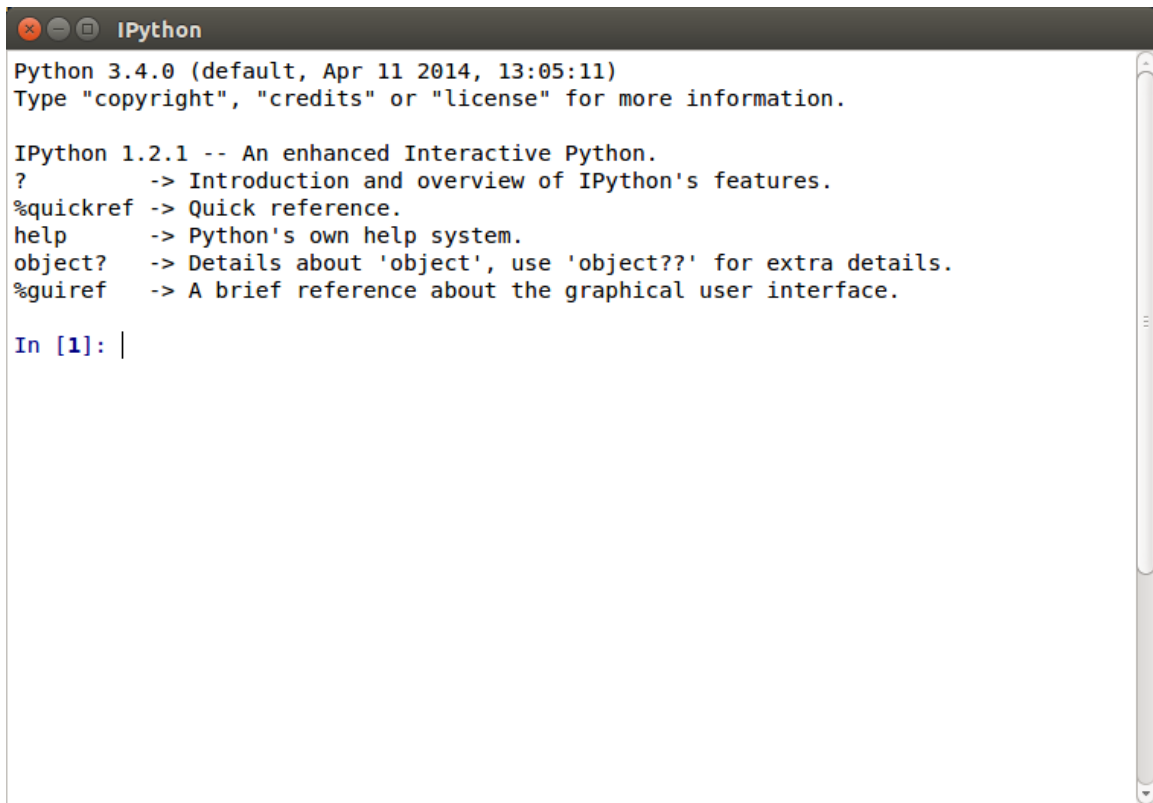
2.5.1 Ipython3

Ipython3 aslında bir programdır. Yaptığı iş bize Python3 ailesinin etkileşimli kabuğunu farklı bir dizayn ile sunmaktır. Tabii ki avantajları yalnızca farklı dizaynı değil, aynı zamanda birtakım özellikleri sayesinde de klasik etkileşimli kabuktan ayrılıyor. Bu yazımda normal Ipython'dansa Ipython3'ün Qt kütüphanesi ile zenginleştirilmiş versiyonunu kullanacağım. Öncelikle aşağıdaki uçbirim komudu ile IPython3-Qt programını sisteme kuralım:

```
1 ~$sudo apt-get install ipython3-qtconsole
```

Eğer normal Ipython3 programını kurmak isterseniz kullanacağınız komut şu şekildedir:

```
1 ~$sudo apt-get install ipython3]
```

A screenshot of an IPython terminal window. The window title is "IPython". The text inside the window reads: "Python 3.4.0 (default, Apr 11 2014, 13:05:11)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
%gui -> A brief reference about the graphical user interface.

In [1]: |". The window has a standard macOS-style title bar with red, yellow, and green buttons on the left and a vertical scrollbar on the right.

```
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
%gui -> A brief reference about the graphical user interface.

In [1]: |
```

Şekil 2: Ipython3G

Tabii bu komutlar yalnızca Debian tabanlı sistemler içindir. Kurulum sonucu programı çalıştırdığınızda aşağıdaki gibi bir görüntü elde edeceksiniz:

Henüz Python hakkında pek bir şey öğrenmiş olmadığımızdan programın önemli özelliklerini gösteremiyorum. Ancak seri boyunca bu etkileşimli kabuğu kullanacağımızdan birçok özelliği belki de fark etmeden öğrenmiş olacaksınız. Siz bu süre boyunca klasik etkileşimli kabuğu kullanabileceğiniz gibi 'Bpython3' yazılımına kullanabilirsiniz.

2.6 Python ile Programlama Mantığı

Python programlama zevkli bir programlama tipi olsa da bazı katı kurallara sahiptir. Bunlardan şimdilik bilmemiz gereken girinti meselesidir.

2.6.1 Girintiler

Python programlama dilinde, yorumlayıcının dikkat ettiği hususlardan biri de girintilerdir. Python'da her ':' işaretinden sonraki satır girintili olmak zorundadır. Yorumlayıcı ':' işaretinden önceki kısmın devam edip etmediğini girintileri denetleyerek anlar. Diğer birçok programlama dilinde bu görev '{...}' işaretleri ile karşılanır. Girinti dediğimiz bilgisayardaki boşluk tuşuna dört kez bastığımızda ya da bunun yerine bir kez 'tab' tuşuna bastığımızda oluşan boşluktur. İnternette ya da bazı kaynaklarda buna 'beyaz boşluk' denildiğini görebilirsiniz.

Bir Python programında bir kez dört boşluk yoluyla girinti yöntemini kullandıysanız bir daha 'tab' tuşunu kullanamazsınız. Aynı şekilde bir kez 'tab' tuşu ile girinti oluşturduysanız bir daha dört boşluk yöntemini kullanamazsınız.

Python, klasik etkileşimli kabuğa göre daha gelişmiş olduğundan ':' gördüğü yerde dört boşluk yoluyla girintileme işini bizim yerimize yapıyor. Ancak bu denemeyi klasik etkileşimli kabukta yapıyorsanız dört boşluğu kendiniz eklemelisiniz. Klasik etkileşimli kabuk sizden girinti beklediğini '>>>' yerine '...' göstererek ifade eder.

Gelin şimdi bu öğrendiğimizi Python programında deneyelim. Programı açıp 'In[1]:' kısmına sadece ':' işareti koyalım. Göreceksiniz ki program dört boşluğu kendisi atayarak imleci bir alt satırda dört boşluk öteye sürüklemiştir:

2.6.2 Yorum Satırları

Python'da ve diğer tüm programlama dillerinde yorum cümleleri çok önemlidir. Yorum cümleleri programlama kodlarına dikkat etmeden günlük dil ile programın içinde yazılmış cümlelerdir. Tabii ki biz bu cümleleri istediğimiz gibi programın içine yerleştirirsek yorumlayıcı şaşırarak bize hata mesajı döndürecektir. Python'da yorum satırlarının başına '#' işareti ekleriz. Bu sayede yorumlayıcının bu satırları hiç dikkate almadan geçmesini sağlarız. Ancak bu satırlar diğer kullanıcının sizin kaynak kodlarınıza bakarken nasıl düşündüğünüzü anlamasını sağlayacaktır. Bunu sadece diğer kişiler için yapmayız. Bazen kendi kodlarımızı bizim bile anlayamadığımız olur. Bu zamanlarda yorum satırları yardımımıza koşar.

Programlama insanlara düşünmeyi öğretir. İyi bir programcı yazdığı kodlarla belli olur. Bilgisayara bir işi onlarca farklı kod yazımı ile yaptırabilirsiniz. Ancak bir sol kulağınıza sol elinizle erişmek var bir de sol kulağınıza başınızın üstünden sağ elinizle erişmek var. Yani bir işi on satırla da yapabilirsiniz üç beş satırla da. Bu yolları yerine göre, en doğru yolla kullanmak için iyi bir programcı olmalısınız. Madem programlamada bir iş için pek çok farklı yol var; biz de kullandığımız yolu belirtmek isteriz. Bunun için de yorum satırlarını kullanırız.

Umarım yorum satırlarının gerekliliğini şimdi daha iyi anlatmışımdır. Şimdi dilerseniz yorum satırını Python programında deneyelim. Python programını açıp içine şu satırları yazalım:

```
1 #Bu bir yorum satırıdır.
```

Gördüğünüz gibi etkileşimli kabuk hiçbir şey yapmadan bir alt satıra geçti. Aslında çok boş bir iş gibi değil mi?

```
IPython
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.
%gui?      -> A brief reference about the graphical user interface.

In [1]: :
...:     boşluk bırakıldı ve bu satıra 4 boşlukla başlandı|
```

Şekil 3: Girinti

2.6.3 Nesne - Değişken Tanımları

Python %100 nesne yönelimli bir programlama dili olduğundan, Python'da her şeye bir isim vermeliyiz. Ya da şöyle ifade edelim Python'da her şey bir nesnedir (Şimdilik deęişkendir diyebiliriz.). Ancak bizim bu nesnelere ulaşabilmemiz için ona bir isim vermemiz gerekir. Hemen bir örnek:

```
1 A = 5
```

Bu satırı Ipython'da yazdığınız zaman 5 sayısını 'A' olarak isimlendirmiş oluyorsunuz. Denemek için bu satırı yazdıktan sonra kod olarak sadece 'A' yazın. Sonuç olarak 5 değeri gelmiş olmalıdır.

Bu nesneleri teker teker atayabileceğimiz gibi birkaç nesneyi bir satır kod ile de atayabiliriz.

```
1 A,B = 5, 3
```

Bu şekilde 'A' ve 'B' olarak iki değeri atamış olduk. Dilerseniz Ipython'a 'A' ve 'B' kodlarını satır satır yazarak ne değerler döndürdüklerine bakabilirsiniz.

Python büyük ve küçük harflere duyarlıdır. Yani Ipython'a şu satırları yazdığınızda:

```
1 B = 5
2 A = 3
3 b = 7
```

ve 'b', 'B' kodlarını tek tek denediğinizde farklı değerler döndüğünü göreceksiniz.

Eğer daha önce tanımlamadığınız bir nesnenin ismini Python yorumlayıcısına sorarsanız, yani henüz 'a' adlı bir nesne tanımlamadan Ipython'a 'a' diye bir satır yazarsanız, Python size bir hata mesajı gösterecektir:

```
1 NameError
2 Traceback (most recent call last)
3 <ipython-input-1-60b725f10c9c> in <module>()
4 ----> 1 a
5
6 NameError: name 'a' is not defined
```

Bu hata mesajında bize 'a' değişkeninin henüz tanımlı olmadığını bildiriyor. Diğer konuya geçmeden önce Python'da değer tanımlama kurallarından bahsedeyim:

- Python'da nesne tanımlarken özel karakter olarak yalnızca '_' işaretini kullanabilirsiniz.
- Python nesne adlarının başında sayı karakterleri bulunmaz.
- Nesne adları, Python'da daha önce belirlenmiş karakterlerden olamaz. (ÖR: 'def', 'print', 'in')

2.7 Programlama yaparken karşılaşılan hatalar

Bu bölümde öncelikle genel anlamda programlama hatalarından bahsedeceğim. Çünkü şu an için bilginiz buna yetiyor. Python bilginiz yeterli düzeye ulaştığında bu hataları yine fark ettirmeden size tüm detaylarıyla anlatmış olacağım.

- **Sözdizimi Hataları:** Programlama esnasında yapılan yanlış yazımdır. Genelde programlama kuralları katıdır. Yapılan küçük hatalar bile programın tamamının ya da bir kısmının çalışmamasına sebep olabilir. Bu hatalar genellikle kolayca çözülür çünkü derleyiciler ya da yorumlarımız bize bu hataları ve nerede olduklarını belirtir.
- **Anlamsal Hatalar:** Programlama yaparken sözdizimi hataları kadar sık görülmeyen ancak çözülmesi çok daha zor olabilen hata tipidir. Bu hata tipi derleyiciler ya da yorumlayıcılar tarafından sözdizimi hataları gibi hata detaylarını belirtmezler. Hatta hatayı hiç anlayamazlar bile. Şimdi bu hatayı bildiklerimiz dahilinde anlamaya çalışalım;

Örneğin sadece toplama işlemi yapabildiğim bir hesap makinesi yazılımı yapıyorum. Her şey tamamlandı. Artık tek yapmam gereken programı çalıştırmak. Evet program bir konsol programı yani program uçbirim üzerinden çalışıyor. Neyse... Programı çalıştırdım. Hatasız çalıştı! Bu ilk programlarımdan olduğu için ve gerçekten bir iş yaptığı için çok sevdim, âdeta havalara uçtum, hiç beklemeden bu programı kız arkadaşımın götürdüm. Tahminimce çok şaşıracağı ve nasıl yaptığımı soracaktı. Çünkü herkesin bildiği gibi (aslında yanlış biliyorlar) bilgisayar programları çok zor yapılıyordu. Eee bu bir de konsol uygulaması; insanların gözüne daha bi' garip, daha bi' zor geliyor. Zaten yazılımdan ballandıra ballandıra bahsedecektim, bayağı etkilenirdi.. Bunları düşünürken evlerine vardım. Alelacele bilgisayarı açtım Python'u kurdum. Bu süre boyunca kız arkadaşımın bu işi ballandıra ballandıra anlatma işini büyük oranda bitirmiştim. Oldukça merak etmiştim. Evet programı çalıştırdım karşımızda şöyle bir ekran:

```
>>>Serhat'ın kız arkadaşı için oluşturduğu programa hoşgeldiniz..!
```

```
Lütfen toplanacak sayılardan birincisini girin: |
```

Hemen artistik tavırlarla 1 yazdım. Program benden 2. sayıyı istedi ona da 1 yazdım. 'Enter' tuşuna basmamla rezaleti fark ettim. Göz kapaklarım çekilmişti, ağzım istemsiz açılmıştı. Rezil olmuşum. Kız arkadaşımın kahkaları ağır çekimde gözümün önüne geliyordu. Ekranda aynen şunlar yazıyordu:

```
>>>Serhat'ın kız arkadaşı için oluşturduğu programa hoşgeldiniz..!
```

```
>>>Lütfen toplanacak sayılardan birincisini girin: 1
```

```
>>>Alınan sayı = 1
```

```
>>>Lütfen toplanacak sayılardan ikincisini girin: 1
```

```
>>>Alınan sayı =1
```

```
>>> 1 + 1 = 11
```

Evet. Biraz komik, biraz hüzünlü, garip bir hikâye ile size bu konuyu böyle anlatan ilk kişiyim diye düşünüyorum. Burada Serhat'ın düştüğü hataya aynı biçimde olmasa da ben de okulda düştüm. Açıkçası ben de çok gülmüş-tüm. Burada Serhat'ın doğru sonucu alamaması bir anlamsal hatadır. Program hata vermedi ancak istenilen sonucu da vermedi. Sanırım bu hata tipini de anladınız. Serhat'ın yapıdığı programı ve doğrusunu serimizin ikinci sayısında beraber göreceğiz.

- **Çalıştırma Zamanı Hataları:** Bu hata tipi ile genelde kullanıcılar karşılaşır. Programın farklı zamanlarda farklı davranış göstermesidir. Örneğin döviz kurlarını her açıldığında internetten alan bir program internet olmadığında hata verip kapanıyor, bu bir hatadır ve bu hatanın adı çalıştırma zamanı hatasıdır. İyi bir programcı bu hataları henüz yazım sürecindeyken görmelidir, önlemli almalıdır. Programcılar çok çok iyi olsa bile bu hataların tamamını saptayamayabilir. İşte açık kaynak kodlu yazılımların avantajı burada görülür. Açık kaynak kodlu yazılımları kullanan kişiler bu hataları fark edip hataların yerini saptayabilir. Ancak açık kaynak kodlu olmayan yazılımlarda bu hatalar fark edilse de ancak programcıya haber verip programcının hatayı çözmesini bekleyebilir.

2.8 Âdettendir... 'Merhaba Dünya'(Hello World)

Başka programlama dillerini öğrendiyseniz ya da öğreniyorsanız bilirsiniz. Bu işler hep 'Hello World' programlarıyla başlar. Bu bir âdettir. Hatta kültüredür bile diyebiliriz. Aslında çoğu kez bu işin eşğine kadar geldik, defalarca bu programı yazdık, çalıştırdık. Ancak artık ciddi anlamda programlamaya başlayacağımızdan bunuda kısacık bi' görelim bakalım. Bu programı yapmak Python dilinde tek satırdan ibarettir:

```
1 print("Merhaba Dünya")
```

İPython3 yazılımını ya da herhangi bir Python3 yorumlayıcısını açıp yukarıdaki satırı yazdığınızda ekranda şöyle bir çıktı alırız:

```
1 Merhaba Dünya
```

Şu anda bilinçli olarak bir Python programı yaptınız. Artık kendinize programcı deyin. :)

Bu programı yaparken bir yapı kullandık: 'print'. Gelin, bu yapı programlamada ne işimize yarayacak bir bakalım.

2.9 Print(Yazdır) Fonksiyonu:

Öncelikle başlıkta kullandığımız ifadeyi bir şekilde açıklığa kavuşturalım. **Fonksiyon**, eğer hatırlıyorsanız, matematik konusundaki fonksiyonlara çok benzer. Hatırlamanız için bir örnek vereyim:

$$f(x) = 2x + 5$$

ifadesi bir matematiksel fonksiyondur. Burada vereceğimiz x değeri, fonksiyonun işlemleri süzgecinden geçirilip yeni değer alır. Yani x'e 3 değeri verince bu 3 değeri fonksiyonun içine girip önce 2 ile çarpılacak, daha sonra 5 eklenecektir. *Yeni x sayımız 11 olacaktır.*

Programlama fonksiyonları da bu şekilde işler. İçinde verilen değeri belirlenen işlemlerden geçirir ve sonucu gösterir (Bazen göstermeyebilir de.).

İşte print yapısı da bir fonksiyondur. Merhaba Dünya programımızda x yerine print'e, "Merhaba Dünya" değeri verildi. print bu verilen değeri biz görmesek de birtakım işlemlerden geçirip bizim için ekrana yazdırdı. Yani print fonksiyonun görevi verilen değerleri ekrana yazdırmak. Bu arada bu değer dediğimiz ifedelere parametre denir. Yani fonksiyonun içine yazdığımız şeyler birer parametredir.

Print fonksiyonuna vereceğimiz parametre bir yazı dizisi ise bu yazı dizisini tırnak işareti("''") içine almamız gerekir. Aksi takdirde hata alırız. Aldığımız hataya bi' göz atalım:

```
1 File "<stdin>", line 1
2     print(Merhaba Dünya)
3         ^
4 SyntaxError: invalid syntax
```

Python bize bir sözdizimi hatası yaptığımızı belirtti. Şimdi lpython3 yazılımını açıp birkaç deneme yapalım:

```
1 print("Bu bir parametredir.")
```

```
1 print("Print bir fonksiyondur")
```

```
1 print("Print aldığı parametreleri ekrana yazar")
```

```
1 print("Verilen parametre bir yazı dizisi ise tırnak işareti kullanmak gerekir")
```

```
1 print("1234567890*—!'^+%&/()¼½=?::;|}][{ $#")
```

Gördüğünüz gibi print ona ne parametre verirsek ekrana yazdırdı, tabii tırnak içine aldığımız takdirde. Ancak aklınıza geldi mi bilmiyorum ama biz tırnak işareti kullanmamız gerektiğinde ne yapacağız?

Örneğin ""İzmir üzerine dünyada bir şehir daha yoktur!" diyorlar. (Yahya Kemal Beyatlı)" satırını ekrana yazdırmak için ne yapabiliriz? Önce bir tırnak içine koyarak deneyelim:

```
1 print("""İzmir üzerine dünyada bir şehir daha yoktur! " diyorlar. (Yahya Kemal Beyatlı)""")
```

Aldığımız hata:

```
1 File "<stdin>", line 1
2     print("""İzmir üzerine dünyada bir şehir daha yoktur! " diyorlar. (Yahya Kemal Beyatlı)""")
3         ^
4 SyntaxError: invalid syntax
```

Bu hatayı yukarıda print ifadesine yazı dizilerini tırnak içinde vermeyince de gördük. Sözdizimi hatası...

Aslında işin başından beri yazı dizisi olarak belirttiğim şey Python'da karakter dizisi olarak adlandırılır. Karakter dizileri tırnak içine aldığımız karakterlerdir. Yani birtakım karakteri tırnak içine aldığımızda karakter dizisi elde ediyoruz. Karakter dizilerini serinin öteki sayılarında detaylıca göreceğiz. Ancak karakter dizilerini oluşturmanın yöntemlerine ufaktan bir göz atalım.

Karakter dizilerini dört şekilde oluşturabilirsek de biz ufakça göz gezdireceğimiz için üç şekline bakacağız.

Bunlardan birincisi bildiğimiz çift tırnak işareti. Bunun üzerinde zaten durduk.

İkinci yol ise tek tırnak işareti ('). Bu ifade yoluyla da karakter dizisi oluşturabiliriz. Çift tırnak işaretinde farkı yoktur. Yalnızca karmaşayı önlemede yardımcı olur.

```
1 print('Merhaba Dünya')
```

Üçüncü yol ise (""") yöntemidir. Yani karakter dizisinin başına ve sonuna üç adet çift tırnak işareti koymaktır.

```
1 print("""Merhaba Dünya""")
```

Bu ifade şöyle de kullanılabilir:

```
1 print(''Merhaba Dünya''')
```

Bunlardan hangisi ile karakter dizisi oluşturmaya başlarsanız karakter dizini yine sadece seçtiğiniz yöntemin işareti ile bitirebilirsiniz. Yani bu aşağıdaki satırları çalıştırdığınızda hata almayacaksınız demektir:

```
1 print(' "İzmir üzerine dünyada bir şehir daha yoktur! " diyorlar. (Yahya Kemal Beyatlı) ')
```

```
1 print(" 'İzmir üzerine dünyada bir şehir daha yoktur!' diyorlar. (Yahya Kemal Beyatlı)")
```

Siz de diğer kombinasyonlarla deneyip aldığınız hataların sebeplerini araştırabilirsiniz.

Print fonksiyonuna sadece bir parametre vermek zorunlu değildir. Parametreler virgüllerle ayrılır. Print fonksiyonuna birden fazla karakter dizinini parametre olarak verebiliriz. Yani şu ifade yanlış değildir:

```
1 print('Merhaba Dünya', "Ben Geldim")
```

Print fonksiyonu parametreleri bu şekilde alınca araya bir boşluk bırakıp ekrana yazar. Bu sayede print fonksiyonu sonsuz parametre alabilir.

Nesneden print'e, Python çağırıyor:

Python'da nasıl nesne tanımlayacağımızı öğrenmiştik. Ama örneklerde sadece sayı kullanmıştık. Esasında bir nesne Python'da her veri tipinin ismi olabilir.

Python'da nesneleştirdiğimiz her şey bilgisayarın belleğine kaydedilir. Bu sebeple büyük işlemler yapılması gereken işlerde, o işi bir nesneye verirsek iş belleğe kaydedilmiş olur. Gerektiğinde bilgisayar ona daha hızlı ulaşmış olur.

Bu ufak bilgiden sonra bir iki tane nesneleme örneği vereyim:

```
1 a = 2
2 b = 'Nesne'
3 c = '2 kalem'
4 A = 5
```

Aslında nesneleme işlemi eşitlemektir. Yani yukarıdaki kodları şöyle çevirebiliriz:

```
a Python için 2 demektir. b ise Python için 'Nesne' demektir.
```

Madem bunlar aynı şeyler, biz bunları print içinde uygulayabiliriz. Örneklere bakalım ve IPython'3'te uygulayalım:

```
1 a = 5
2 b = 4
3 c = 'Pazardan'
4 d = 'Mareketten'
5
6 print(c, b, 'poşet', d, a, 'poşet')
```

Bu şekilde print fonksiyonuna nesnelere de ekleyebileceğimizi gördük. Aynı zamanda print fonksiyonuna sayı değeri verebileceğimizde görmüş olmalısınız. 'a' ve 'b' nesnelere tırnak işaretleri arasında olmayan sayı içeriyor. Buna rağmen print hata vermeden çalıştı. İşte size yeni bir veri tipi. Python'da sayı veritipini normal şekilde belirtebiliriz. Print, sayı veritipini de başarıyla ekrana yazdırabilir.

Print parametreleri:

Print fonksiyonu sadece sizin vereceğiniz parametrelerle sınırlı değildir. Bu fonksiyon bazı öntanımlı parametrelerde barındırır. Öntanımlı derken siz değiştirmedığınız sürece ona verilen değeri kullanan parametrelerden bahsediyorum. Bunlardan iki tanesinden bahsedelim hemen:

1. **sep:** Bu parametreyi anlatırken önceden verdiğimiz bir örnek üzerinden gideyim.

```
1 print('Merhaba Dünya', "Ben Geldim")
```

Örneğimiz buydu. Bunu çalıştırdığımızda alacağımız çıktı:

```
1 Merhaba Dünya Ben Geldim
```


olur. Ancak fark ettiyseniz Python hiçbir şey demeden araya boşluk bırakıp yazdırdı verdiğimiz parametreleri. Ama bu yanlış. Eğer programlama yapıyorsak bu durumu değiştirebiliyor olmalıyız.

Evet, değiştirebiliyoruz. Bunu 'sep' parametresi ile yapıyoruz. Yani sep parametresine verdiğimiz değeri print, ekrana yazdıracağımız değerlerin arasına koyuyor. Mmm karışık gibi.. Örnek olsa daha iyi olacak:

```
1 a = "Fizik"
2 b = "Kimya"
3 c = "Biyoloji"
4
5 print(a, b, c)
```

Python3'te bunun aynısını yaptığımızda aldığımız çıktı:

```
1 Fizik Kimya Biyoloji
```

Ancak biz bunu şu şekilde yazarsak daha iyi olur sanki:

```
1 a = "Fizik"
2 b = "Kimya"
3 c = "Biyoloji"
4
5 print(a, b, c, sep='—')
```

Bu kez alacağımız çıktı:

```
1 Fizik—Kimya—Biyoloji
```

2. **end:** Bu parametre ise yine bizim haberimiz olmadan bir şeyler yapılmasını engelleyecek. Ya da başka bir deyişle bir tür farkındalık yaratacak.

Print fonksiyonunun son yaptığı şey alt satıra geçmektir. Yani tüm işini bitirip alt satıra geçer. Bu durum bazı durumlarda sinir bozucu olabilir. Şimdi bir örnek vereyim ve örneği açıklayıp yeni konumuza geçelim:

```
1 print('Nane', 'Nane', sep=" — ", end=" ")
```

Eğer bu örneği IPython3 te çalıştırdıysanız hiçbir değişiklik gözlememiş bulunmaktasınız. IPython3 yapısı gereği bu tür değişiklikleri esgeçer. Siz bu örneği bir de normal yorumlayıcıdan deneyin ve değişikliği görün.

Burada yapmış olduğumuz şey print fonksiyonuna 'işin bitince alt satıra geçme bir boşluk bırakıp devam et' demek. Ve bu söylemi print'e end = " " ifadesiyle anlattık. İşimiz bitince bir kez daha nane yazdırmak isteseydik böyle bir kod yazabilirdik:

```
1 print('Nane', 'Nane', sep=" — ", end="Nane")
```

2.10 Input (Giriş) Fonksiyonu

Buraya kadar print fonksiyonu hakkında çok şey öğrendik. Ancak print ile yapabildiklerimiz tek taraflı şeylerdi. Yani programı kullanan kişi programa bir şey yaptıramıyordu.

Şimdi öğreneceğimiz fonksiyon sayesinde kullanıcıdan da veri alabileceğiz. Fonksiyonumuzun adı '**input**'.

Input fonksiyonuna giriş adına bu örneği IPython'a yazalım:

```
1 input("Giriş: ")
```

Bu örneği çalıştırdığınızda karşınıza bir Giriş: yazısı çıkar. Eğer hiçbir şeye basmadan devam ettiyseniz bir OUT[..] satırı ve karşısında " karakterlerini görürsünüz.

Bu kodu tekrar çalıştırıp Giriş: yazısına herhangi bir şey yazın. OUT[..] çıktısında o yazdığınız karakterleri göreceksiniz.

Yani input fonksiyonunun içine yazdığımız karakter dizisi önce ekranda görünür. Ardından Python bizden cevap bekler. Cevabı verip 'enter' tuşuna basınca input fonksiyonu bu cevabı döndürür. Input fonksiyonu için birkaç örnek:

```
1 input('Giriniz:')
2 input('Lütfen Giriş Yapın: ')
3 input("Merhaba sen kimsin ")
```

Input'tan nesneye, beni Python gönderdi:

Input fonksiyonu kullanıcıdan bir değer alıp bu değeri döndürür. Eğer biz bu fonksiyonu yukarıdaki gibi kullanırsak alınan veriyi bir daha kullanamayız. Aldığımız veriyi kullanmak için input fonksiyonunu bir nesneye atmalıyız:

```
1 a = input("a'nın değeri: ")
```

Bu kodu çalıştırdığımızda bize a'nın değerini sordu, ancak verdiğimiz cevabı bize göstermedi. Çünkü Python o cevabı a nesnesine yükleyip belleğe gönderdi. Şimdi aynı ekrana 'a' yazdığınızda verdiğiniz değer dönecektir. Yani şunu demek istiyorum:

```
1 a = input("a'nın değeri: ")
2 a
```

E beni de input gönderdi:

Bir önceki bölümde input fonksiyonunu nesneye atadık. Şimdi bu atadığımız nesneyi biraz kullanalım.

Öncelikle Python yorumlayıcısını açalım ve bir input nesnesi oluşturalım örneğin:

```
1 Giriş = input('Lütfen Kullanıcı Adınızı Girin: ')

```

Bu input nesnesini artık istediğimiz gibi kullanabiliriz. Örneğin :

```
1 print(Giriş)
```

Tabii bu çok kolay oldu. Şimdi öğrendiklerimizle daha kullanışlı bir şeyler yapalım:

```
1 Hoşgeldiniz_Yazısı = """
2 Hoş Geldiniz...
3 Bu program Python diliyle konsol
4 üzerinden yazılmıştır."""
5
6 print(Hoşgeldiniz_Yazısı)
7 K_Adı = input('Lütfen Kullanıcı Adını Girin: ')
8 print(K_Adı, 'Adlı Üye Başarıyla Giriş Yaptı!')
```

Bu kısımda (" " ") işaretinin kullanımına dikkatinizi çekmek isterim. Eğer karakter dizilerini bu şekilde kullanırsak girmiş olduğumuz yeni satırlarda karakter dizisine dahil olacaktır. Bu sadece bu şekilde karakter dizisi tanımlamanın özelliğidir. Geri kalan şekillerde yeni satıra geçme işi '\n' ifadesiyle yapılır.

2.11 Hesap Makinesi Python!

Python programlama diliyle aritmetik işlemler de çok rahatlıkla yapılabilir. Python programlama dilinde aritmetik işlemler yalnızca sayılar ve ondalık sayılar ile yapılır. Python'da sayılar ve ondalıklı sayılar farklı veritipleridir.

Ondalıklı sayılarda tam kısım ve kesir kısım Türkçe kullanımının aksine '.' ile ayrılır.

Python'da 4 işlem çok basitçe yapılabilir:

```
1 a = 2 + 3
2 b = 5 - 6
3 c = 3 * 2
4 d = 3 / 1
```

Hemen çıktıları görelim:

```
1 >>> a
2 5
3
4 >>>b
5 -1
6
7 >>>c
8 6
9
10 >>>d
```

Yani burada programlama adına yaptığımız tek şey sonuçları nesnelendirmek. Şimdi bu konuda daha karışık bir örnek vereyim. Siz bu satırları kopyalayarak lpython3'te çalıştırabilirsiniz:

```
1 x = 3
2 deę = (((2*(x+4)+3) / 7 - 4) + 129/8) * x + 5)/((x*2)/7)
3 print(deę)
```

Sonucun 56.770833333333336 sayısı olması gerekli.

Bu aritmetik operatörler dışında 2 operatör daha bilmemiz bize yarar sağlar.

- **%**: Bölme işleminden kalanı verir

```
1 8 % 3
```

- **__**__:** Sayının kuvvetini almaya yarar. Şimdi çıktıları ile birlikte örnek yapalım:

```
1 >>>3 ** 2
2 9
3
4 >>>2 ** 10
5 1024
```

2.12 Karakter Dizilerinin Toplama ve Çarpma Özelliği:

Konunun başında aritmetik işlemler sadece sayılar için kullanılır dedik. Aslında aritmetik işlemler olarak kullanılmasa da çarpma işlemi ve toplama işlemi sembolleri diğer veri tiplerinde de kullanılabilir. Şimdilik bunlardan karakter dizileri için olan kısmı işleyeceğiz.

Python'da karakter dizilerine kendi aralarında toplama işlemi uygulamak bu dizileri birleştirmek demektir. Hemen bir örnek:

```
1 >>> a = 'bir'+ 'sen'+ 'bir'+ 'ben'+ 'birde'+ 'bebek'
2 'birsenbirbenbirdebebek'
```

Daha işe yarar bir örnek:

```
1 >>> a = 'Sen'
2 >>> b = 'Ben'
3 >>> c = 've'
4 >>> print(a + ' ' + c + ' ' + b)
5 'Sen ve Ben'
```

Aritmetik işlemleri karakter dizileri için kullandığımız bir diğer nokta çarpma sembolüdür. Çarpma sembolünün bir tarafında karakter dizisi varken öteki tarafında sayı bulunur.

```
1 >>> "Karakter " * 3
2 'Karakter Karakter Karakter '
```

Şimdi bu öğrendiklerimizi kullanalım:

```
1 >>> print('Merhaba Dünya')
2 >>> a = 'Adın Ne'
3 >>> s = '?'
4 >>> Yazı = a + s*3
5 >>> print(Yazı)
6 Adın Ne???
```

3 Kapanış

Bu seferlik bu bölümün sonuna geldik. İlk sayımdı. Büyük heyecan ve büyük tecrübesizliğim vardı. Bu sayıda öğrendiğimiz programlama mantığı ve Python programlama diline giriş bölümleri umarım hayatınıza katkı yapmıştır, yapacaktır...

Normalde bu seri boyunca her sayının sonunda sorular kısmı olacak. Ancak bu sayıda yeterli bilgimiz olmadığından bu kısmı es geçiyorum. Bu bilgilerle yapmanız gereken bol tekrar etmektir. Deneme yanılma yöntemi ile yeni şeyler öğrenmektir.

Bu süre boyunca Ubuntu-tr forumu olarak her zaman size destek vermeye hazırız.

Ayrıca bir sonraki sayıya kadar örnek ve bazı açıklamaları Ubuntu-tr forumunun Sudo-Python konusundan bulabilirsiniz.