Birden Fazla Satırı Kapsayan Tek Bir İfade:

**Bazen ifadeler oldukça karmaşık hale gelir. Önerilen Python stili, satırları boşluklar, sekmeler ve diğer boşluk karakterleri dahil olmak üzere 80 karakterle sınırlamaktır ve bu, programlama dünyasında yaygın bir sınırdır. İşte satırlar çok uzadığında veya anlaşılır olması için bölmek istediğinizde yapmanız gerekenler.**

**Bir ifadeyi birden fazla satıra bölmek için iki şeyden birini yapmanız gerekir:**

**1. Satır sonunun parantez içinde olduğundan emin olun.**

**2. Ters eğik çizgi olan satır sonu karakterini kullanın \.**

**Satır sonu karakterinin bölme sembolü (/) değil ters eğik çizgi (\) olduğuna dikkat edin.**

**İşte her ikisine de örnekler:**

**>>> (2 +**

**. . . 3)**

**5**

**>>> 2 + \**

**. . . 3**

**5**

**Nasıl SyntaxError almadığımıza dikkat edin. Örneklerimizdeki her üç noktalı komut istemi, bir ifade girmenin ortasında olduğumuzu gösterir; bunları kodu güzel bir şekilde sıralamak için kullanırız. Noktaları, normal >>> komut isteminde büyüktür işaretlerini yazdığınızdan daha fazla yazmazsınız ve IDLE kullanıyorsanız, onları hiç görmezsiniz.**

**İşte daha gerçekçi (ve daha lezzetli) bir örnek: diyelim ki kurabiye pişiriyoruz. Yazarlar Kanada'da yaşıyorlar ve Celsius kullanıyorlar ama bizim yemek kitaplarımız Fahrenheit kullanıyor. Fırınımızı önceden ısıtmanın ne kadar süreceğini merak ediyoruz. İşte gerçeklerimiz:**

**- Oda sıcaklığı 20 santigrat derecedir.**

**- Fırın kontrollerimiz Santigrat kullanıyor ve fırın her 20 derecede bir ısınıyor.**

**Bir dakika.**

**- Yemek kitabımız Fahrenheit kullanıyor ve fırını önceden 350 dereceye ısıtmamızı söylüyor.**

**t Fahrenheit derecesini t Santigrat derecesine şu şekilde dönüştürebiliriz:**

**(t - 32) \* 5 / 9.**

**Bu bilgiyi problemimizi çözmeye çalışmak için kullanalım.**

**>>> oda\_sıcaklığı\_c = 20**

**>>> pişirme\_sıcaklığı\_f = 350**

**>>> fırın\_ısıtma\_hızı\_c = 20**

**>>> fırın\_ısıtma\_zamanı = (**

**... ((pişirme\_sıcaklığı\_f - 32) \* 5 / 9) - oda\_sıcaklığı\_c) / \**

**... fırın\_ısıtma\_hızı\_c**

**>>> fırın\_ısıtma\_zamanı**

**7.833333333333333**

**Fena değil - ön ısıtma için sekiz dakikadan biraz az. oven\_heating\_time değişkenine atama deyimi üç satırdan oluşur. İlk satır açık bir parantezle biter, bu nedenle satır sonu karakterine ihtiyacımız yoktur. İkinci satır parantezin dışında biter, bu nedenle satırsonu karakterine ihtiyacımız vardır. Üçüncü satır atama ifadesini tamamlar.**

**Bunu okumak hala zor. Bir ifadeyi bir sonraki satırda devam ettirdikten sonra, daha anlaşılır hale getirmek için istediğimiz kadar girinti yapabiliriz (Tab tuşuna basarak veya boşluk çubuğuna bir sürü basarak):**

**>>> fırın\_ısıtma\_zamanı = (**

**... ((pisirme\_sicakligi\_f – 32) \* 5 / 9) - oda\_sicakligi\_c) / \**

**... fırın\_ısıtma\_hızı\_c**

**Çıkarma işlemindeki iki alt ifadenin nasıl tam olarak hizalandığına bakın:**

**>>> fırın\_ısıtma\_zamanı = (**

**... ((pisirme\_zamanı\_f – 32) \* 5 / 9) -**

**... oda\_sıcalıgı\_c) / \**

**... fırın\_ısıtma\_hızı\_c**

**Önceki örnekte, girinti kullanarak ifadeyi netleştirdik. Ancak, pişirme sıcaklığını Celsius'a dönüştürdükten sonra ısıtma süresini hesaplayarak bu süreci daha da açık hale getirebiliriz:**

**>>> oda\_sıcaklıgı\_c = 20**

**>>> pisirme\_sıcaklıgı\_f = 350**

**>>> pisirme\_sıcaklıgı\_c = (pisirme\_sıcaklıgı\_f – 32) \* 5 / 9**

**>>> fırın\_ısıtma\_hızı\_c = 20**

**>>> fırın\_ısıtma\_zamanı = ( pisirme\_zamanı\_c – oda\_sıcaklıgı\_c) / \**

**... fırın\_ısıtma\_hızı\_c**

**>>> fırın\_ısıtma\_zamanı**

**7.833333333333333**

**Buradan çıkarılacak mesaj, iyi adlandırılmış geçici değişkenlerin kodu çok daha anlaşılır hale getirebileceğidir.**

Kodu Tanımlama:

**Programlar oldukça karmaşık olabilir ve genellikle binlerce satır uzunluğundadır. Kodun bazı kısımlarını açıklayan bir yorum yazmak yararlı olabilir, böylece siz veya başka biri okuduğunda anlam açık olur.**

**Python'da # karakteri ile karşılaşıldığında, Python satırın geri kalanını yok sayacaktır. Bu, İngilizce cümleler yazmanıza olanak sağlar:**

**>> # Python # sembolü nedeniyle bu cümleyi yok sayar.**

**# sembolü satırdaki ilk karakter olmak zorunda değildir; bir ifadenin sonunda da görünebilir:**

**>>>** **(212 – 32) \* 5 / 9 # 212 Fahrenheit dereceyi Celsius'a çevirin. 100.0**  
  
**Yorumun Python'un nasıl çalıştığını açıklamadığına dikkat edin. Bunun yerine, kodu okuyan insanların kodun neden var olduğunu anlamalarına yardımcı olmak içindir.**  
  
Kodu Okunabilir Hale Getirme:  
  
**İngilizce cümlelerde kelimelerin okunmasını kolaylaştırmak için boşluklar olduğu gibi, Python kodunda da okumayı kolaylaştırmak için boşluklar kullanırız. Özellikle, her ikili operatörden önce ve sonra daima bir boşluk bırakırız. Örneğin, v=4+-2.5/3.6 yerine v = 4 + -2.5 / 3.6 yazarız. Bunun bir fark yaratmayabileceği durumlar vardır, ancak bu üzerinde durmak istemediğimiz bir ayrıntıdır, bu yüzden her zaman yaparız: boşluk varsa okumak neredeyse hiçbir zaman daha zor değildir.**  
  
**Psikologlar, insanların herhangi bir zamanda yalnızca birkaç şeyi takip edebildiğini keşfetmiştir (Psikolojiyi Değiştiren Kırk Çalışma [Hoc04]). Programlar oldukça karmaşık hale gelebildiğinden, değişkenleriniz için ne işe yaradıklarını hatırlamanıza yardımcı olacak isimler seçmeniz önemlidir. id1, X2 ve blah, gelecek hafta programınıza bakmak için geri döndüğünüzde size hiçbir şey hatırlatmayacaktır: bunun yerine celsius, average ve final\_result gibi isimler kullanın.**

**Diğer çalışmalar beyninizin otomatik olarak şeyler arasındaki farklılıkları fark ettiğini göstermiştir - aslında bunu yapmasını engellemenin bir yolu yoktur. Sonuç olarak, bir metin parçasında ne kadar çok tutarsızlık varsa, okumak o kadar uzun sürer. (Eğer bu şekilde düzenlenmiş olsaydı bu kitabı okumanızın ne kadar süreceğini bir düşünün). Bu nedenle değişkenler için tutarlı isimler kullanmak da önemlidir. Bir yerde bir şeye maksimum diyorsanız, başka bir yerde max\_val demeyin; max\_val adını kullanıyorsanız, maxVal adını da kullanmayın, vb.**  
  
  
**Bu kurallar o kadar önemlidir ki birçok programlama ekibi, tıpkı gazetelerin ve kitap yayıncılarının başlıkların nasıl büyük harfle yazılacağını ve bir listedeki son öğeden önce virgül kullanılıp kullanılmayacağını belirtmesi gibi, üyelerinden kullandıkları dil için bir stil kılavuzuna uymalarını ister. İnternette programlama stil rehberi (https://www.google.com/search?q=programming+style+guide) şeklinde arama yaparsanız yüzlerce örneğe bağlantı bulabilirsiniz. Bu kitapta, Python için http://www.python.org/dev/peps/pep-0008/ adresindeki stil kılavuzunu takip ediyoruz.**

**Ayrıca pek çok insanın kod için “en iyi” stilin ne olduğunu tartışarak saatler harcadığını da keşfedeceksiniz. Bazı sınıf arkadaşlarınızın (ve eğitmenlerinizin) da bu konuda güçlü fikirleri olabilir. Eğer varsa, onlara inançlarını desteklemek için hangi verilere sahip olduklarını sorun. Güçlü fikirlerin ciddiye alınması için güçlü kanıtlar gerekir.**

Bu Bölümün Amacı:  
Bu bölümde aşağıdakileri öğrendiniz:  
  
- İşletim sistemi, bilgisayarınızın donanımını diğer programlar adına yöneten bir programdır. Bir yorumlayıcı ya da sanal makine, işletim sisteminin üzerine oturan ve programlarınızı sizin için çalıştıran bir programdır. Python kabuğu bir yorumlayıcıdır, Python ifadelerinizi işletim sisteminin anladığı dile çevirir ve sonuçları geri çevirir, böylece onları görebilir ve kullanabilirsiniz.  
  
- Programlar deyimlerden ya da talimatlardan oluşur. Bunlar 3 + 4 gibi basit ifadeler ve celsius = 20 gibi atama ifadeleri olabilir (yeni değişkenler oluşturur veya mevcut olanların değerlerini değiştirir). Python'da başka birçok deyim türü vardır ve bunları kitap boyunca tanıtacağız.  
  
  
- Python'daki her değer, kendisine hangi işlemlerin uygulanabileceğini belirleyen belirli bir türe sahiptir. Sayıları temsil etmek için kullanılan iki tür int ve float'tır. Kayan noktalı sayılar gerçek sayıların yaklaşık değerleridir.  
  
- Python bir ifadeyi yüksek öncelikli operatörleri düşük öncelikli operatörlerden önce uygulayarak değerlendirir. Alt ifadelerin etrafına parantez koyarak bu sırayı değiştirebilirsiniz.  
  
- Python her değeri bilgisayar belleğinde saklar. Bir değer içeren bellek konumuna nesne denir.  
  
- Değişkenler atama deyimleri çalıştırılarak oluşturulur. Eğer bir değişken daha önceki bir atama deyimi nedeniyle zaten mevcutsa, Python yeni bir değişken oluşturmak yerine bu değişkeni kullanır.  
  
- Değişkenler, değerlerin bellek adreslerini içerir. Değişkenlerin değerlere atıfta bulunduğunu söyleriz.  
  
- Değişkenlere, ifadelerde kullanılabilmeleri için önce değer atanmalıdır.