**3.2.4 Yerleşik Kapsam**

72

3 özyineleme

Python'daki son kapsam Yerleşik kapsamdır. Bir tanımlayıcı, bir modül içindeki iç içe geçmiş kapsamlarda bulunamazsa ve global kapsamda tanımlanmamışsa, Python yerleşik tanımlayıcıları inceleyerek orada tanımlanıp tanımlanmadığına bakar. Örneğin, int tanımlayıcısını düşünün. Aşağıdakileri yazarsanız:

X = int("6")

Python önce yerel kapsamda int ‘in o yerel kapsamda bir fonksiyon veya değişken olarak tanımlanıp tanımlanmadığına bakar. İnt yerel kapsamda bulunamazsa, Python en içteki yerel kapsamdan başlayarak ve dışarı doğru çalışarak tüm kapsayan kapsamları arar. Herhangi bir kapsayan kapsamda bulunamazsa, Python global kapsamda int tanımlayıcısını arar. Orada da bulunamazsa, Python yerleşik kapsamdaki int sınıfı veya tipini bulur. Bu açıklama ile, yerleşik kapsamda zaten var olan tanımlayıcıları kullanmamanız gerektiği şimdi daha net olmalıdır. İnt gibi bir tanımlayıcı kullanırsanız, Python yerel veya kapsayan bir kapsamda int'i bulacağı için yerleşik kapsamdaki int'i kullanamazsınız.

**3.2.5 LEGB**

Mark Lutz, Learning Python [6] adlı kitabında, Python programlarındaki kapsam kurallarını LEGB kısaltmasıyla anlattı. Bu kısaltma, Yerel, Kapsayan, Global ve Yerleşik anlamına gelen Local, Enclosing, Global ve Built-In kelimelerinin baş harflerinden oluşur ve Python'daki kapsam kurallarını ezberlemenize yardımcı olabilir. Kısaltmadaki harflerin sırası önemlidir. Python yorumlayıcısı bir programda bir tanımlayıcıyla karşılaştığında, önce yerel kapsamı arar, ardından içten dışa doğru tüm kapsayan kapsamları, ardından global kapsamı ve son olarak yerleşik kapsamı arar.

**3.3 Çalışma Zamanı Yığını ve Yığın**

Geçen bölümde öğrendiğimiz gibi, her fonksiyonun parametreleri ve gövdesi Python programı içinde bir kapsam tanımlar. Bir fonksiyonun yerel kapsamı içinde tanımlanan parametreler ve değişkenler bilgisayarın RAM’ı içinde bir yerde saklanmalıdır. Python RAM’ı Çalışma Zamanı Yığını ve Yığın olarak adlandırılan iki bölüme ayırır. Çalışma zamanı yığını, bir kafeteryadaki tepsi yığınına benzer. Çoğu kafeteryada bu tepsileri tutan bir cihaz vardır. Tepsi yığını yeterince kısaldığında, tepsilerin altındaki bir yay tepsileri yukarı kaldırır, böylece güzel bir yükseklikte olurlar. Yığına daha fazla tepsi eklendikçe, bu cihazdaki yay sıkışır ve yığın aşağı doğru itilir. Bilgisayar Bilimlerindeki Yığın, birçok yönden bu tür bir cihaza benzer. Çalışma zamanı yığını bir Aktivasyon Kayıtları yığınıdır.

Python yorumlayıcısı, bir fonksiyon çağrıldığında çalışma zamanı yığınına bir aktivasyon kaydı ekler. Bir fonksiyon geri döndüğünde Python yorumlayıcısı ilgili aktivasyon kaydını çalışma zamanı yığınından kaldırır. Python, yerel kapsamda tanımlanan tanımlayıcıları bir etkinleştirme kaydında saklar. Fonksiyon çağrıldığında, yeni bir kapsam yerel kapsam haline gelir. Aynı zamanda çalışma zamanı yığınına yeni bir aktivasyon kaydı eklenir. Bu yeni aktivasyon kaydı, yeni yerel kapsam içinde tanımlanan tüm değişkenleri tutar. Bir fonksiyon geri döndüğünde, ilgili aktivasyon kaydı çalışma zamanı yığınından kaldırılır.

73

Yığın Bellek (Heap), RAM’de tüm nesnelerin saklandığı alandır. Bir nesne oluşturulduğunda heap'te bulunur. Çalışma zamanı yığını asla nesne içermez. Nesnelere yapılan referanslar çalışma zamanı yığını içinde saklanır ve bu referanslar yığındaki nesnelere işaret eder. Şekil 3.2'deki programı düşünün. Python yorumlayıcısı programın 23. ve 24. satırlarını çalıştırırken, çalışma zamanı yığını Şekil 3.3'teki gibi görünür. Çalışma zamanı yığınında üç aktivasyon kaydı vardır. Çalışma zamanı yığınına eklenen ilk aktivasyon kaydı modül içindir. Modül ilk çalışmaya başladığında, Python yorumlayıcısı modülü yukarıdan aşağıya doğru gözden geçirir ve modül kapsamındaki tüm değişken tanımlarını modül için aktivasyon kaydına ekler. Bu programda bu, 3.14159 değerine PI referansından oluşuyordu.

Ardından, modülün sonunda if ifadesi main fonksiyonunu çağırdı. Bu, Python yorumlayıcısının ana fonksiyon için aktivasyon kaydını eklemesine neden oldu. Ana fonksiyon içinde tanımlanan değişkenler arasında historyOfPrompts, history OfOutput, rString, r ve val bulunur. Bunların her biri ana fonksiyonun aktivasyon kaydında görünür.

Ana fonksiyon çalışmaya başladığında getInput fonksiyonunu çağırır. Bu çağrı gerçekleştiğinde, fonksiyon çağrısı için bir aktivasyon kaydı eklenmiştir. Bu aktivasyon kaydı prompt ve x değişkenlerini içeriyordu. Bu aktivasyon kaydı şekilde görünmüyor çünkü programın 23. ve 24. satırlarını çalıştırdığımızda Python yorumlayıcısı getInput fonksiyonundan çoktan dönmüştü. Yorumlayıcı fonksiyon çağrısından döndüğünde, ilgili aktivasyon kaydı çalışma zamanı yığınından kaldırılmıştır.

Son olarak, program *26. satırda showOutput* fonksiyonunu çağırır ve fonksiyonun yürütülmesi başlar. ShowOutput çağrıldığında showOutpout fonksiyon çağrısı için bir aktivasyon kaydı çalışma zamanı yığınına eklenmiştir. Sadece val değişkenini içeren bu kapsam için yerel referanslar, bu fonksiyon çağrısı için aktivasyon kaydında saklanmıştır.

Bu örnek programı Wing ya da başka bir IDE kullanarak çalıştırabilirsiniz. Bunun için kod Bölüm 20.2'de görünmektedir. Bu programı çalıştırmak için Wing IDE'yi kullandığınızda, programı herhangi bir noktada durdurabilir ve çalışma zamanı yığınını inceleyebilirsiniz. Örneğin, Şekil 3.4 Wing'i bu programı çalıştırırken göstermektedir. Programı durdurmak için 24. satırda bir kesme noktası ayarlanmıştır. Wing IDE penceresinin altındaki sekme Yığın Verisini gösterir. Bu çalışma zamanı yığınıdır.

Yığın Verileri sekmesinin hemen altında, şu anda showOutput() işlevini görüntüleyen bir kombinasyon kutusu vardır: Scope.py, satır 24. Bu kombinasyon kutusu, o anda görüntülenmekte olan aktivasyon kaydı arasından seçim yapmanızı sağlar. Farklı bir aktivasyon kaydı seçerseniz, içeriği Wing IDE'de doğrudan onun altında görüntülenecektir.

metin, diyagram, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

3 Özyineleme

74

**Şekil 3.3** Çalışma Zamanı Yığını ve Yığın

Burada önemli bir not düşülmelidir. Şekil3.4, showOutput fonksiyonunda historyOfOutput'u yerel bir değişken olarak göstermektedir. Gerçekte durum böyle değildir, çünkü historyOfOutput referansı showOutput fonksiyonunun yerel kapsamında tanımlanmamıştır. Ancak, Python'un uygulanma şekli nedeniyle, bu değişken için referans showOutput için aktivasyon kaydında görünebilir çünkü bu kapsamdan referans alınmaktadır. Ancak, showOutput aktivasyon kaydındaki historyOfOutput referansı ile ana aktivasyon kaydındaki historyOfOutput adlı referans aynı nesneyi işaret ettiğinden gerçek bir zarar oluşmaz. Dikkat edilmesi gereken önemli nokta, Wing IDE'nin historyOfOutput değişkenini bu aktivasyon kaydında yerel bir değişken olarak göstermekte haklı olduğudur, çünkü bu Python'un uygulamasının bir yansımasıdır ve Wing IDE 101'deki bir hatadan kaynaklanmamaktadır.

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu  
**Şekil 3.4** Çalışma Zamanı Yığınını Gösteren Wing IDE

75

3.4 Özyinelemeli Bir Fonksiyon Yazma

**3.4 Yinelemeli Fonksiyon Yazma**

Özyinelemeli bir fonksiyon basitçe kendini çağıran bir fonksiyondur. Özyinelemeli bir fonksiyon yazmak gerçekten çok basittir, ancak elbette gerçekten ilginç bir şeyler yapan özyinelemeli fonksiyonlar yazmak istersiniz. Buna ek olarak, eğer bir fonksiyon kendi kendini çağırmaya devam ederse asla bitmez. Aslında, bir bilgisayarda çalıştırıldığında biterdi çünkü bir fonksiyonu her çağırdığınızda, çalışma zamanı yığınına bir aktivasyon kaydı eklendiğini öğrendik. Özyinelemeli bir fonksiyon kendini tekrar tekrar çağırmaya devam ederse,

sonunda çalışma zamanı yığınını doldurur ve böyle bir programı çalıştırırken yığın taşması hatası alırsınız.

3 Özyineleme

76

Özyinelemeli bir fonksiyonun sonsuza kadar çalışmasını veya çalışma zamanı yığınının taşmasını önlemek için, her özyinelemeli fonksiyonun bir temel durumu olmalıdır, tıpkı tümevarımsal bir ispatın bir temel durumu olması gerektiği gibi. Tümevarımsal ispatlar ile özyinelemeli fonksiyonlar arasında birçok benzerlik vardır. Özyinelemeli bir fonksiyonun temel durumu, fonksiyon özyinelemeli olarak çağrılmadan önce yazılmalıdır.

Şimdi, özyinelemeli bir fonksiyonun nasıl çalıştığını anlamak ilk başta biraz zordur. Aslında, özyinelemeli bir fonksiyonun nasıl çalıştığını anlamak o kadar da önemli değildir. Özyinelemeli fonksiyonlar yazarken, nasıl çalıştığından çok ne yaptığını düşünmek isteriz. Özyinelemeli fonksiyonların nasıl çalıştığına dair çok fazla kafa yormaya gerek yoktur, ancak aslında bu bile biraz pratikle çok daha kolay hale gelecektir.

Özyinelemeli bir fonksiyon yazarken uymanız gereken dört kural vardır. Bu kurallar tartışmaya açık değildir ve özyinelemeli fonksiyonunuzun eninde sonunda bitmesini sağlayacaktır. Bu kuralları ezberler ve takip etmeyi öğrenirseniz, kısa sürede özyinelemeli fonksiyonlar yazmaya başlayacaksınız. Kurallar şunlardır:

1. Fonksiyonunuzun adına ve işini tamamlaması için ona iletilmesi gereken argümanlara ve fonksiyonun hangi değeri döndürmesi gerektiğine karar verin.
2. Önce özyinelemeli fonksiyonunuz için temel durumu yazın. Temel durum, bir değer döndürerek özyinelemeli işlevdeki çok basit bir durumu ele alan bir if durumudur.
3. Son olarak, fonksiyonu son çağrı yapıldığında fonksiyona aktarılan parametrelerden bir şekilde daha küçük olan bir argüman veya argümanlarla özyinelemeli olarak çağırmalısınız. Küçülen argüman veya argümanlar, temel durumunuzda incelediğiniz argüman veya argümanlarla aynıdır.
4. Somut bir örneğe bakın. Özyinelemeli fonksiyonunuzla denemek için bazı değerler seçin. Son adımda yaptığınız özyinelemeli çağrının çalıştığına emin olun. Bu özyinelemeli çağrının sonucunu alın ve fonksiyonunuzun döndürmesini istediğiniz sonucu oluşturmak için kullanın. Bu sonucu nasıl oluşturacağınızı görmenize yardımcı olması için somut örneği kullanın. Başlangıç için çok basit bir örnek yapacağız. Son bölümde aşağıdakileri kanıtlamıştık.

yazı tipi, beyaz, taslak, diyagram içeren bir resim

Açıklama otomatik olarak oluşturulduDolayısıyla, ilk n tamsayının toplamını hesaplamak istersek, Bölüm 3.4.1'de gösterildiği gibi bir Python programı yazabiliriz.

**3.4.1 Tamsayılar Toplamı**

**metin, makbuz, yazı tipi, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu**