Introduction

Bir fonksiyonun girdi değerini değiştirdiğinizde genellikle farklı bir çıktı elde ettiğinizi daha önce görmüştünüz. Örneğin, herhangi bir sayıya beş ekleyen ve sonucu döndüren bir add\_five() fonksiyonunu ele alalım. Bu durumda add\_five(7) 12 (=7+5) çıktısını, add\_five(8) ise 13 (=8+5) çıktısını döndürecektir. Girdi ne olursa olsun, fonksiyonun gerçekleştirdiği eylemin her zaman aynı olduğunu unutmayın: her zaman beş ekler.

Ancak bunun yerine, girdiye bağlı bir eylem gerçekleştiren bir fonksiyona ihtiyacınız olabilir. Örneğin, girdi 10'dan küçükse üç, 10 veya daha büyükse sekiz ekleyen bir add\_three\_or\_eight() fonksiyonuna ihtiyacınız olabilir. Bu durumda add\_three\_or\_eight(1) 4 (=1+3) döndürürken, add\_three\_or\_eight(11) 19 (=11+8) döndürür. Bu durumda, fonksiyonun gerçekleştirdiği eylem girdiye göre değişir.

Bu derste, fonksiyonlarınızın çalışma şeklini değiştirmek için koşulları ve koşullu ifadeleri nasıl kullanacağınızı öğreneceksiniz.

# Conditions

Programlamada koşullar, Doğru veya Yanlış olan ifadelerdir. Python'da koşulları yazmanın birçok farklı yolu vardır, ancak en yaygın koşul yazma yöntemlerinden bazıları iki farklı değeri karşılaştırmaktır. Örneğin, 2'nin 3'ten büyük olup olmadığını kontrol edebilirsiniz.

print(2 > 3)

False

Python bunu False olarak tanımlar, çünkü 2, 3'ten büyük değildir.

Değişkenlerin değerlerini karşılaştırmak için koşulları da kullanabilirsiniz. Bir sonraki kod hücresinde, var\_one 1 değerine, var\_two ise 2 değerine sahiptir. Koşullarda, var\_one'ın 1'den küçük olup olmadığını (yani False) ve var\_two'nun var\_one'dan büyük veya ona eşit olup olmadığını (yani True) kontrol ederiz.

var\_one = 1

var\_two = 2

print(var\_one < 1)

print(var\_two >= var\_one)

False

True

Koşulları oluşturmak için kullanabileceğiniz yaygın sembollerin listesi için aşağıdaki tabloya bakın.

|  |  |
| --- | --- |
| **Symbol** | **Meaning** |
| == | equals |
| != | does not equal |
| < | less than |
| <= | less than or equal to |
| > | greater than |
| >= | greater than or equal to |

Önemli Not: İki değerin eşit olup olmadığını kontrol ederken = işaretini değil, == işaretini kullandığınızdan emin olun.

var\_one==1, var\_one değerinin 1 olup olmadığını kontrol eder, ancak

var\_one=1, var\_one değerini 1 olarak ayarlar.

# Conditional statements

Koşul ifadeleri, fonksiyonunuzun çalışma şeklini değiştirmek için koşullar kullanır. Bir koşulun değerini kontrol ederler ve koşul True (Doğru) olarak değerlendirilirse, belirli bir kod bloğu çalıştırılır. (Aksi takdirde, koşul False (Yanlış) ise, kod çalıştırılmaz.)

Buna dair birkaç örneği aşağıdaki bölümlerde göreceksiniz.

## **"if" statements**

En basit koşullu ifade türü "eğer" ifadesidir. Bunun bir örneğini aşağıdaki evaluate\_temp() fonksiyonunda görebilirsiniz. Fonksiyon, giriş olarak vücut sıcaklığını (Santigrat cinsinden) kabul eder.

Başlangıçta, mesaj "Normal sıcaklık" olarak ayarlanır.

Daha sonra, sıcaklık > 38 ise (örneğin, vücut sıcaklığı 38°C'nin üzerindeyse), mesaj "Ateş!" olarak güncellenir. Aksi takdirde, sıcaklık > 38 ise (Yanlış ise), mesaj güncellenmez.

Son olarak, fonksiyon tarafından mesaj döndürülür.

def evaluate\_temp(temp):

*# Set an initial message*

message = "Normal temperature."

*# Update value of message only if temperature greater than 38*

if temp > 38:

message = "Fever!"

return message

Bir sonraki kod hücresinde, sıcaklığın 37°C olduğu fonksiyonu çağırıyoruz. Mesaj "Normal sıcaklık", çünkü sıcaklık bu durumda 38°C'den düşük (sıcaklık > 38 ise False olarak değerlendirilir).

print(evaluate\_temp(37))

Normal temperature.

Ancak sıcaklık 38°C'den büyük olduğundan 39°C ise mesaj "Ateş!" olarak güncellenir.

print(evaluate\_temp(39))

Fever!

İki düzeyde girinti olduğunu unutmayın:

İlk düzeyde girinti, bir fonksiyon içindeki kod bloğunu her zaman girintilememiz gerektiği içindir.

İkinci düzeyde girinti ise, "if" ifadesine ait kod bloğunu da girintilememiz gerektiği içindir. (Göreceğiniz gibi, "elif" ve "else" ifadeleri için de kod bloklarını girintilememiz gerekecek.)

Return ifadesi "if" ifadesinin altında girintilenmediği için, temp > 38 değeri True veya False olsun, her zaman çalıştırılır.

## **"if ... else" statements**

Bir ifadenin False olması durumunda kodu çalıştırmak için "else" ifadelerini kullanabiliriz. "if" ifadesinin altındaki kod, ifade True ise çalıştırılır ve "else" ifadesinin altındaki kod, ifade False ise çalıştırılır.

def evaluate\_temp\_with\_else(temp):

if temp > 38:

message = "Fever!"

else:

message = "Normal temperature."

return message

Bu evaluate\_temp\_with\_else() fonksiyonu, evaluate\_temp() fonksiyonuna eşdeğer bir davranışa sahiptir.

Bir sonraki kod hücresinde, sıcaklığın 37°C olduğu bu yeni fonksiyonu çağırıyoruz. Bu durumda, sıcaklık > 38 ise False olarak değerlendirilir, bu nedenle "else" ifadesinin altındaki kod çalıştırılır ve Normal sıcaklık mesajı döndürülür.

print(evaluate\_temp\_with\_else(37))

Normal temperature.

Önceki fonksiyonda olduğu gibi, "if" ve "else" ifadelerinden sonra kod bloklarını girintiliyoruz.

## **"if ... elif ... else" statements**

Birden fazla koşulun doğru olup olmadığını kontrol etmek için "elif" ("else if"in kısaltması) kullanabiliriz. Aşağıdaki fonksiyon:

Öncelikle sıcaklığın 38'den büyük olup olmadığını kontrol eder. Bu doğruysa, mesaj "Ateş!" olarak ayarlanır.

Mesaj daha önce ayarlanmamışsa, fonksiyon sıcaklığın 35'ten büyük olup olmadığını kontrol eder. Bu doğruysa, mesaj "Normal sıcaklık" olarak ayarlanır.

Daha sonra, hala bir mesaj ayarlanmamışsa, "else" ifadesi mesajın "Düşük sıcaklık" olarak ayarlanmasını sağlar. Mesaj yazdırılır.

"Elif"i şöyle bir şey olarak düşünebilirsiniz... "Tamam, önceki koşul (örneğin, sıcaklık > 38) yanlıştı, öyleyse bu yeni koşulun (örneğin, sıcaklık > 35) doğru olup olmadığını kontrol edelim!"

def evaluate\_temp\_with\_elif(temp):

if temp > 38:

message = "Fever!"

elif temp > 35:

message = "Normal temperature."

else:

message = "Low temperature."

return message

Aşağıdaki kod hücresinde, temp > 38 False, temp > 35 ise True olduğundan, kodu "elif" ifadesi altında çalıştırıyoruz. Bu kod çalıştırıldığında, fonksiyon "else" ifadesini atlar ve mesajı döndürür.

evaluate\_temp\_with\_elif(36)

'Normal temperature.'

Son olarak, sıcaklığın 35°C'nin altında olduğu bir durumu deneyelim. "if" ve "elif" ifadelerindeki koşulların her ikisi de False olarak değerlendirildiğinden, "else" ifadesinin içindeki kod bloğu çalıştırılır.

evaluate\_temp\_with\_elif(34)

'Low temperature.'

# Example - Calculations

Şimdiye kadarki örneklerde, değişkenlerin değerlerinin nasıl ayarlanacağına karar vermek için koşullu ifadeler kullanıldı. Ancak, farklı hesaplamalar yapmak için de koşullu ifadeler kullanabilirsiniz.

Bir sonraki örnekte, yalnızca iki vergi diliminin olduğu bir ülkede yaşadığınızı varsayalım. 12.000'den az kazanan herkes %25, 12.000 veya daha fazla kazanan herkes ise %30 vergi öder. Aşağıdaki fonksiyon, ne kadar vergi borcu olduğunu hesaplar.

def get\_taxes(earnings):

if earnings < 12000:

tax\_owed = .25 \* earnings

else:

tax\_owed = .30 \* earnings

return tax\_owed

Bir sonraki kod hücresi fonksiyonu kullanır.

ana\_taxes = get\_taxes(9000)

bob\_taxes = get\_taxes(15000)

print(ana\_taxes)

print(bob\_taxes)

2250.0

4500.0

Her durumda, get\_taxes() fonksiyonunu çağırıyoruz ve döndürülen değeri bir değişkenin değerini ayarlamak için kullanıyoruz.

ana\_taxes fonksiyonu için, 9.000 kazanan bir kişinin borçlu olduğu vergileri hesaplıyoruz. Bu durumda, kazançlar 9000 olarak ayarlandığında get\_taxes() fonksiyonunu çağırıyoruz. Dolayısıyla, kazançlar < 12000 ise True (Doğru) ve tax\_owed fonksiyonu .25 \* 9000 olarak ayarlanıyor. Ardından tax\_owed değerini döndürüyoruz.

bob\_taxes fonksiyonu için, 15.000 kazanan bir kişinin borçlu olduğu vergileri hesaplıyoruz. Bu durumda, kazançlar 15000 olarak ayarlandığında get\_taxes() fonksiyonunu çağırıyoruz. Dolayısıyla, kazançlar < 12000 ise False (Yanlış) ve tax\_owed fonksiyonu .30 \* 15000 olarak ayarlanıyor. Ardından tax\_owed değerini döndürüyoruz. Başka bir örneğe geçmeden önce, giriş bölümündeki add\_three\_or\_eight() fonksiyonunu hatırlayın. Giriş olarak bir sayı alır ve giriş 10'dan küçükse üç, değilse sekiz ekler. Bu fonksiyonu nasıl yazacağınızı tahmin edebilir misiniz? Cevabınızı bulduktan sonra, çözümü görmek için aşağıdaki "Gizli kodu göster" butonuna tıklayın.

# Example - Multiple "elif" statements

Şimdiye kadar, "elif" ifadesini bir fonksiyonda yalnızca bir kez gördünüz. Ancak kullanabileceğiniz "elif" ifadesi sayısında bir sınırlama yoktur. Örneğin, bir sonraki kod bloğu, bir çocuğa verilecek ilaç dozunu (mililitre cinsinden) kilosuna (kilogram cinsinden) göre hesaplar.

Not: Bu fonksiyon tıbbi tavsiye olarak kullanılmamalıdır ve sahte bir ilacı temsil eder.

def get\_dose(weight):

*# Dosage is 1.25 ml for anyone under 5.2 kg*

if weight < 5.2:

dose = 1.25

elif weight < 7.9:

dose = 2.5

elif weight < 10.4:

dose = 3.75

elif weight < 15.9:

dose = 5

elif weight < 21.2:

dose = 7.5

*# Dosage is 10 ml for anyone 21.2 kg or over*

else:

dose = 10

return dose

Bir sonraki kod hücresi işlevi çalıştırır. Çıktının sizin için anlamlı olduğundan emin olun!

Bu durumda, "if" ifadesi False idi ve tüm "elif" ifadeleri, weight < 15,9'a ulaşana kadar False olarak değerlendirilir; bu da True'dur ve dose değeri 5 olarak ayarlanır.

Bir "elif" ifadesi True olarak değerlendirilip kod bloğu çalıştırıldığında, işlev kalan tüm "elif" ve "else" ifadelerini atlar. Bunları atladıktan sonra geriye sadece dose değerini döndüren return ifadesi kalır.

Elif ifadelerinin sırası burada önemlidir! İfadelerin yeniden sıralanması çok farklı bir sonuç döndürecektir.

print(get\_dose(12))

5