Booleans

Python'da bool adında bir değişken türü vardır. İki olası değeri vardır: True (Doğru) ve False (Yanlış).

x = True

print(x)

print(type(x))

True

<class 'bool'>

Kodumuza doğrudan True veya False eklemek yerine, genellikle Boolean operatörlerinden Boolean değerleri alırız. Bunlar, evet/hayır sorularını yanıtlayan operatörlerdir. Aşağıda bu operatörlerden bazılarını inceleyeceğiz.

## **Comparison Operations**

| Operation | Description |  | Operation | Description |
| --- | --- | --- | --- | --- |
| a == b | a equal to b |  | a != b | a not equal to b |
| a < b | a less than b |  | a > b | a greater than b |
| a <= b | a less than or equal to b |  | a >= b | a greater than or equal to b |

def can\_run\_for\_president(age):

*"""Can someone of the given age run for president in the US?"""*

*# The US Constitution says you must be at least 35 years old*

return age >= 35

print("Can a 19-year-old run for president?", can\_run\_for\_president(19))

print("Can a 45-year-old run for president?", can\_run\_for\_president(45))

Can a 19-year-old run for president? False

Can a 45-year-old run for president? True

Karşılaştırmalar çoğu zaman umduğunuz gibi çalışır

3.0 == 3

True

Ama bazen aldatıcı olabiliyorlar

'3' == 3

False

Karşılaştırma operatörleri, neredeyse sınırsız sayıda matematiksel testi ifade etmek için daha önce gördüğümüz aritmetik operatörlerle birleştirilebilir. Örneğin, bir sayının tek olup olmadığını, 2'li modülün 1 döndürüp döndürmediğini kontrol ederek kontrol edebiliriz:

def is\_odd(n):

return (n % 2) == 1

print("Is 100 odd?", is\_odd(100))

print("Is -1 odd?", is\_odd(-1))

üIs 100 odd? False

Is -1 odd? True

Karşılaştırma yaparken = yerine == kullanmayı unutmayın. n == 2 yazarsanız, n'nin değerini soruyorsunuz demektir. n = 2 yazarsanız, n'nin değerini değiştiriyorsunuz demektir.

## **Combining Boolean Values**

Boole değerlerini, "ve", "veya" ve "değil" gibi standart kavramları kullanarak birleştirebilirsiniz. Aslında, bunu yapmak için kullanılan kelimeler şunlardır: ve, veya ve değil.

Bunlarla can\_run\_for\_president fonksiyonumuzu daha doğru hale getirebiliriz.

def can\_run\_for\_president(age, is\_natural\_born\_citizen):

*"""Can someone of the given age and citizenship status run for president in the US?"""*

*# The US Constitution says you must be a natural born citizen \*and\* at least 35 years old*

return is\_natural\_born\_citizen **and** (age >= 35)

print(can\_run\_for\_president(19, True))

print(can\_run\_for\_president(55, False))

print(can\_run\_for\_president(55, True))

False

False

True

Çabuk, bu ifadenin değerini tahmin edebilir misin?

True **or** True **and** False

True

Bunu cevaplamak için, işlem sırasını bulmanız gerekir.

Örneğin, ve işareti veya işaretinden önce değerlendirilir. Bu yüzden yukarıdaki ilk ifade True'dur. Soldan sağa doğru değerlendirseydik, önce True veya True'yu (ki bu True'dur) hesaplar, ardından bu sonucun ve'sini False ile alır ve nihai değeri False olarak verirdik.

Öncelik sırasını ezberlemeye çalışabilirsiniz, ancak daha güvenli bir yol, bolca parantez kullanmaktır. Bu, hataları önlemeye yardımcı olmakla kalmaz, aynı zamanda kodunuzu okuyan herkes için niyetinizi daha net hale getirir.

Örneğin, aşağıdaki ifadeyi ele alalım:

prepared\_for\_weather = have\_umbrella **or** rain\_level < 5 **and** have\_hood **or** **not** rain\_level > 0 **and** is\_workday

Bugünkü hava koşullarından güvende olduğumu söylemeye çalışıyorum...

eğer şemsiyem varsa...

ya da yağmur çok şiddetli değilse ve bir kapüşonum varsa...

aksi takdirde, yağmur yağmadığı ve iş günü olmadığı sürece sorun yok.

Ama Python kodumu okumak zor olduğu gibi, bir de hatası var. Her iki sorunu da parantez ekleyerek çözebiliriz:

prepared\_for\_weather = have\_umbrella **or** (rain\_level < 5 **and** have\_hood) **or** **not** (rain\_level > 0 **and** is\_workday)

Okunabilirliği artıracağını düşünüyorsanız daha fazla parantez ekleyebilirsiniz:

prepared\_for\_weather = have\_umbrella **or** ((rain\_level < 5) **and** have\_hood) **or** (**not** (rain\_level > 0 **and** is\_workday))

Yukarıda anlatılan 3 parçalı yapıyı vurgulamak için bunu birden fazla satıra da bölebiliriz:

prepared\_for\_weather = (

have\_umbrella

**or** ((rain\_level < 5) **and** have\_hood)

**or** (**not** (rain\_level > 0 **and** is\_workday))

)

# Conditionals

Boole ifadeleri, if, elif ve else anahtar sözcüklerini kullanarak koşullu ifadelerle birleştirildiğinde en kullanışlıdır.

Genellikle if-then ifadeleri olarak da adlandırılan koşullu ifadeler, bir Boole koşulunun değerine göre hangi kod parçalarının çalıştırılacağını kontrol etmenizi sağlar. İşte bir örnek:

def inspect(x):

if x == 0:

print(x, "is zero")

elif x > 0:

print(x, "is positive")

elif x < 0:

print(x, "is negative")

else:

print(x, "is unlike anything I've ever seen...")

inspect(0)

inspect(-15)

0 is zero

-15 is negative

if ve else anahtar sözcükleri diğer dillerde de sıklıkla kullanılır; daha özgün anahtar sözcüğü "else if" ifadesinin kısaltması olan elif'tir. Bu koşul cümlelerinde, elif ve else blokları isteğe bağlıdır; ayrıca, istediğiniz kadar elif ifadesi ekleyebilirsiniz.

Özellikle, ayrı kod bloklarını belirtmek için iki nokta üst üste (:) ve boşluk kullanımına dikkat edin. Bu, bir fonksiyon tanımladığımızda olana benzer - fonksiyon başlığı : ile biter ve sonraki satır 4 boşlukla girintilidir. Girintili olmayan bir satırla karşılaşana kadar, sonraki tüm girintili satırlar fonksiyonun gövdesine aittir ve fonksiyon tanımını sonlandırır.

def f(x):

if x > 0:

print("Only printed when x is positive; x =", x)

print("Also only printed when x is positive; x =", x)

print("Always printed, regardless of x's value; x =", x)

f(1)

f(0)

Only printed when x is positive; x = 1

Also only printed when x is positive; x = 1

Always printed, regardless of x's value; x = 1

Always printed, regardless of x's value; x = 0

## **Boolean conversion**

Nesneleri tam sayılara dönüştüren int() ve nesneleri kayan noktalara dönüştüren float() fonksiyonlarını gördük, bu yüzden Python'da nesneleri bool'lara dönüştüren bir bool() fonksiyonu olduğunu duyduğunuzda şaşırmayabilirsiniz.

print(bool(1)) *# all numbers are treated as true, except 0*

print(bool(0))

print(bool("asf")) *# all strings are treated as true, except the empty string ""*

print(bool(""))

*# Generally empty sequences (strings, lists, and other types we've yet to see like lists and tuples)*

*# are "falsey" and the rest are "truthy"*

True

False

True

False

Boole olmayan nesneleri if koşullarında ve boolean beklenen diğer yerlerde kullanabiliriz. Python bunları dolaylı olarak karşılık gelen boolean değerleri olarak ele alacaktır:

if 0:

print(0)

elif "spam":

print("spam")

spam