ITU ACM Student Chapter Course Program

Introduction to Python

Week 2

Instructor
Zafer Yıldız

Assistants
Serra Bozkurt
Hüseyin Averbek

Week 2

Koşullu Durumlar

Koşul ifadeleri, programlamada en çok başvurulan ifadelerden biridir. Koşul ifadelerinin programın içerisindeki fonksiyonu programın takip edeceği yola karar vermek olarak özetlenebilir.

Bazı durumlarda programın içerisinde iki veya daha fazla değişkenin veya durumun karşılaştırılması gerekebilir. Örneğin email hesabımıza giriş yaparken e-postamızı ve şifremizi gireriz. Şifre girişi yapıldıktan sonra girilen şifre sistemde kayıtlı olan şifreyle karşılaştırılır ve iki şifrenin eşleşmesi durumunda e-posta hesabına erişim sağlanır fakat şifrelerin eşleşmediği durumda hata mesajıyla karşılaşırız. Kendisine verilen ifadelerin doğruluğunu kontrol edip gerçek hayattan örnek vermek gerekirse program içerisinde yol ayrımı yaratan programlama yapılarına **koşullu durumlar** denir.

Python dilinde ise koşullu durum yapısını kurmak için **if-else-elif** anahtar kelimeleri kullanılır.

Bu ifadeleri görmeden önce karşılaştırma operatörlerini aşağıdaki tablo üzerinde inceleyelim:

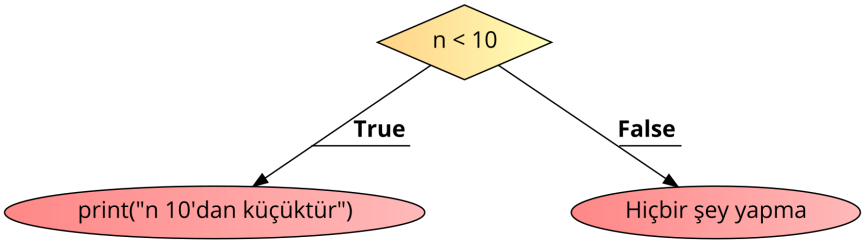
Operatör	Anlamı	Örnek	Örneğin Açıklaması	Değer
<	Küçüktür	3 < 5	3 küçüktür 5	**True**
>	Büyüktür	2 > 7	2 büyüktür 7	**False**
<=	Küçük eşittir	3 <= 3	3 küçük eşittir 3	**True**
>=	Büyük eşittir	4 >= 5	4 büyük eşittir 5	**False**
==	Eşittir	6 == 6	6 eşittir 6	**True**
!=	Eşit değildir	9 != 3	9 eşit değildir 3	**True**

if Bloğu

```
n = int(input("Bir sayı giriniz: "))

if n < 10:                                  #Eğer sayı 10'dan küçükse,
    print("n 10'dan küçüktür.")             #Ekrana "n 10'dan küçüktür" bastır
```

```
Bir sayı giriniz: 15
```



if kullanım algoritması

Yukarıdaki şemada da görüldüğü üzere if'in şartı olarak yazılan "n<10" ifadesinin sağlanması durumunda **if** bloğunun içindeki kod satırı çalıştırılır. Söz konusu koşulun sağlanmadığı durumda ise program **if** bloğunu görmezden gelir. **Örneğin:**

```
yaş = int(input("Yaşınızı giriniz: ")) # Kullanıcı burada yaşını girer.

if yaş < 18:                                          # Program yaşın 18 den␣
 ↪küçük olup olmadığını kontrol eder.
    print("Yaşınız 18'den küçük. Ehliyet alamazsınız") # yaş 18'den küçükse bu␣
 ↪çıktıyı verir.
```

```
Yaşınızı giriniz: 20
```

Yukarıdaki örneğe baktığımızda **if** ifadesinin koşulunun yaş değişkenin 18'den küçük olması olduğunu görüyoruz. Yaş değişkeninin değerinin 18'den küçük olması durumunda program ekrana "Yaşınız 18'den küçük. Ehliyet alamazsınız" çıktısını verir. Fakat 18 veya 18'den büyük bir input girildiği takdirde hiçbir işlem gerçekleşmez.

İkinci bir **if** bloğu yazarak bu durumda da bir işlem gerçekleşmesi sağlanabilir:

```
yaş = int(input("Yaşınızı giriniz: ")) # Kullanıcı burada yaşını girer.

if yaş < 18: # Program yaşın 18 den küçük olup olmadığını kontrol eder.
    print("Yaşınız 18'den küçük. Ehliyet alamazsınız") # yaş 18'den küçükse bu␣
 ↪çıktıyı verir.

if yaş >= 18:
    print("Yaşınız uygun. Ehliyet alabilirsiniz.")
```

```
Yaşınızı giriniz: 22
Yaşınız uygun. Ehliyet alabilirsiniz.
```

Fakat burda iki if bloğunun içindeki şart da doğru olsaydı ikisinin içindeki işlemler de gerçekleştirilirdi. Örneğin:

```python
n = 5

if n < 10:
    print("n 10'dan küçüktür")

if n < 15:
    print("n 15'ten küçüktür")
```

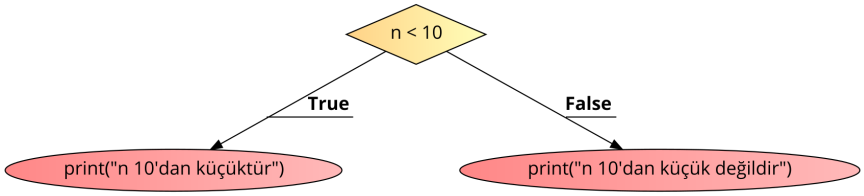
```
n 10'dan küçüktür
n 15'ten küçüktür
```

else Bloğu

```python
n = int(input("Bir sayı giriniz: "))

if n < 10:
    print("n 10'dan küçüktür")
else:
    print("n 10'dan küçük değildir")
```

```
Bir sayı giriniz: 56
n 10'dan küçük değildir
```



else kullanım algoritması

Yukarıdaki şemada da görüldüğü üzere şartın sağlanması durumunda **if** bloğunun içinde bulunan **print("n 10'dan küçüktür")** satırı çalıştırılır. Eğer şart sağlanmıyorsa **else** bloğunun içindeki **print("n 10'dan küçük değildir")** satırı çalıştırılır.

Örneğin bir parola kontrol aşamasını aşağıdaki kod üzerinde inceleyelim:

```python
parola = "123456" # Örnek bir parola değişkeni atandı.

girdi = input("Parolanızı giriniz: ") # Kullanıcıdan parolayı girmesi istendi.
```

```
if girdi == parola: # Parolanın doğruluğu kontrol ediliyor
    print("Parola doğru. Giriş gerçekleştiriliyor...") # Parola doğruysa bu␣
 ↪çıktı ekrana yazdırılır.
else:
    print("Parola yanlış") # Parola doğru değilse bu çıktı ekrana yazdırılır.
```

```
Parolanızı giriniz: 123456
Parola doğru. Giriş gerçekleştiriliyor...
```

Yukarıda da görüldüğü üzere **else** ifadesinin yanına herhangi bir şart ifadesi yazılmaz.

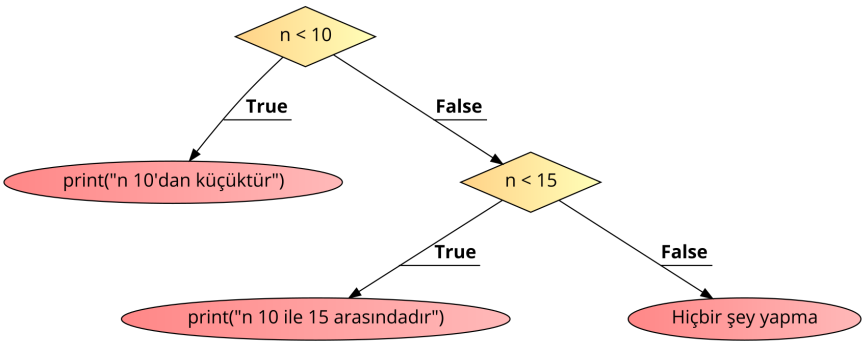
Soru: else bloğunun içine yeni bir if bloğu yazılabilir mi?

elif Bloğu

```
n = int(input("Bir sayı giriniz: "))

if n < 10:
    print("n 10'dan küçüktür.")
elif n < 15:
    print("n 10 ile 15 arasındadır")
```

```
Bir sayı giriniz: 12
n 10 ile 15 arasındadır
```



elif kullanım algoritması

Elif bloğu aşağıda olduğu gibi de yazılabilir:

```
n = 10

if n < 5:
    print("n 5'ten küçüktür")
else:
    if n == 10:
        print("n 10'dur")
```

```
n 10'dur
```

Fakat bu yaklaşımda sadece bir elif bloğu bu şekilde yazılabilir çünkü bir **if** bloğuna yalnızca bir **else** bloğu tekabül edebilir. Python programlama dilinde bu karışıklığı önleme amacıyla **elif** anahtar kelimesi kullanılmaktadır:

```
n = 10

if n < 5:
    print("n 5'ten küçüktür")
elif n == 10:
        print("n 10'dur")
```

```
n 10'dur
```

elif ifadesi birden fazla olabilir:

```
n = 1

if n < 5:
    print("n 5'ten küçüktür")
elif n < 10:
    print("n 5 ile 10 arasındadır")
elif n < 15:
    print("n 10 ile 15 arasındadır")
elif n < 20:
    print("n 15 ile 20 arasındadır")
elif n < 25:
    print("n 20 ile 25 arasındadır")
elif n < 30:
    print("n 25 ile 30 arasındadır")
```

```
n 5'ten küçüktür
```

Birden fazla koşulun sağlandığı bir durumu inceleyecek olursak:

```
n = 5

if n > 5:
    print("n 5'ten büyüktür")
elif n > 10:
    print("n 10'dan büyüktür")
elif n > 15:
    print("n 15'ten büyüktür")
```

Bunun nedeni programın, şartın sağlandığı yerdeki bloğun içine girip diğer şartları kontrol etmemesidir.

if-elif-else bloklarının hepsi birbiriyle ilişkili şekilde de kullanabilir:

```
n = 3

if n < 5:
    print("n 5'ten küçüktür.")
elif n < 10:
    print("n 5 ile 10  arasındadır")
elif n < 15:
    print("n 10 ile 15 arasındadır")
else:
    print("n 15'den küçük değildir")
```

```
n 5'ten küçüktür.
```

Boolean Değerler ve İfadeler

Python'da doğru ve yanlış değerleri saklamak için kullanılan veri tipine bool adı verilmektedir. Boole cebiri tüm modern bilgisayar aritmetiğinin temelidir.

Sadece iki boolean değer vardır: True(doğru) ve False(yanlış). Python case sensitive (büyük-küçük harfe duyarlı) bir dil olduğundan büyük harfle başlamaları önemlidir. true ve false boolean değerler değildir.

```
print(type(True))
```

```
<class 'bool'>
```

```
print(type(true))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-1c2838cf1bc6> in <module>
----> 1 print(type(true))

NameError: name 'true' is not defined
```

Boolean ifadeler, sonucu boolean değer olan ve boolean değer olarak değerlendirilen deyimlerdir.

"==" gibi koşullu durumlar, girdiğimiz bilgilerin doğruluğunu kontrol edip sonuçlar verirler. Verdikleri sonuçlar ise bool tipindedir.

```
print(type(1 == 2))
```

```
<class 'bool'>
```

```
print(5 == 5)
```

```
True
```

```
print(5 == 6)
```

```
False
```

İlk cümlede, karşılaştırılan iki sayı birbirine eşittir, bu yüzden ifade True sonucunu üretir; ikinci cümlede ise 5, 6'ya eşit değildir, bu sebeple False sonucu elde edilir.

Python'da 1 rakamı boolean olarak True değeriyle eşdeğerdir. Aynı şekilde de 0 rakamı False boolean değeriyle eşdeğer kabul edilir.

```
print(1 == True)
```

```
True
```

```
print(0 == True)
```

```
False
```

```
x, y = 3, 2

x == y                # Eğer x, y'ye eşit ise True değeri üretir.
```

```
False
```

```
x != y                # x, y'ye eşit değil ise True değeri üretir.
```

```
True
```

```
x > y                 # x, y'den büyük ise True değeri üretir.
```

```
True
```

```
x < y                 # x, y'den küçük iseTrue değeri üretir.
```

```
False
```

```
x >= y                # x, y'den büyük veya eşit ise True değeri üretir.
```

```
True
```

```
x <= y                # x, y'den küçük veya eşit ise True değeri üretir.
```

```
False
```

Dolayısıyla aslında if-elif-else koşullu ifadelerinin algoritmalarında yapı şu şekildedir:

if "koşullu bir ifade": 1. İfadesinde koşula bakılır. 2. Koşul doğruysa "koşullu bir ifade", True olarak kabul edilir. 3. Şartın True boolean değerine geldiğini gören program o şartın sonucunu gerçekleştirmek için o if koşulunun içine girer. 4. Koşul yanlışsa "koşullu bir ifade", False olarak

kabul edilir. 5. Şartın False boolean değerine geldiğini gören program o şartın sonucunu gerçekleştirmemek için o if koşulunun içine girmez, devam eder. 6. Aşağıdaki muhtemel elif ve else ifadelerinin yanındaki koşullar için birinci aşamadan tekrar kontrol başlar.

```
if 1 == 2:                          # 1 == 2 ifadesi kontrol -> yanlış olduğuna␣
↪karar verildi ->
    print("1, 2'ye eşittir.")       # bu şart satırına geçilmeden devam edildi

elif 1 == 1:                       # 1 == 1 ifadesi kontrol -> doğru olduğuna karar␣
↪verildi ->
    print("1, 1'e eşittir.")        # bu şart satırına geçildi ve içinde verilen␣
↪görev gerçekleştirildi.
```

```
1, 1'e eşittir.
```

Takip eden girintili cümlelere yani iç içe if, elif veya else kullanımına blok adı verilir. İlk girintisiz koşullu ifade bloğun sonunu belirler.

```
yas = int(input("Lütfen yaşınızı giriniz: "))
dogum_tarihi = str(input("Lütfen doğum tarihinizi gg.aa şeklinde giriniz: "))

bugunun_tarihi = "21.09"

if yas < 18:                # yas<18 boolean olarak True ya eşit değilse bu kod␣
↪bloğu yok sayılır, eşitse bu bloğa girilir
    print("Maalesef ehliyet alamıyorsunuz!")              # bloğa girilince␣
↪buradaki işleme başlanır

    if dogum_tarihi == bugunun_tarihi:                    # doğum tarihi␣
↪kontrol edilir
        print("Doğum gününüz kutlu olsun!")               # bugünün tarihine␣
↪eşitse bloğa girilir

    else:                                                 # eşit değilse else␣
↪şartı gerçekleştirilip çıkılır
        print("Lütfen bir sonraki sene tekrar deneyiniz!")
                                                          #buraya ulaşınca ilk␣
↪bloktan çıkılmış olur

elif yas >= 18:                                           # buraya ulaşınca yas␣
↪>= 18 ifadesi
                                                          # True değerine␣
↪eşitse bu bloğa girilir.
    print("Ehliyet alabilirsiniz!")
```

```
Lütfen yaşınızı giriniz: 12
Lütfen doğum tarihinizi gg.aa şeklinde giriniz: 25.01
Maalesef ehliyet alamıyorsunuz!
```

```
Lütfen bir sonraki sene tekrar deneyiniz!
```

Pass

if bloğunun içinde bulunabilecek koşullu cümle sayısında bir sınır yoktur, ama en azından bir ifadenin olması gerekir. Bazen boş if cümlesi yazmak (mesela daha sonra yazmak üzere alan ayırmak) gerekebilir. Bu durumda pass cümlesi kullanılır.

Eğer boş bırakılırsa program hata verir.

Program pass satırına ulaşınca hiçbir işlev gerçekleştirmeden o bloktan çıkar.

```
if 1 == True:        # bu her zaman doğrudur

```

```
  File "<ipython-input-36-a81d02057054>", line 2

    ^
SyntaxError: unexpected EOF while parsing
```

```
if True:             # burada koşullu ifade yazmak yerine direkt boolean değer␣
 ↪yazılmıştır.
                     # bu kullanım da doğrudur
    pass             # henüz ne yazacağımıza karar vermediğimiz kısmı pass ile␣
 ↪hata almaktan kurtardık
```

İç İçe Koşullu İfadeler ve Mantık Operatörleri

Python'da tek if satırında birdan fazla koşulu kontrol etmek için mantık operatörleri kullanılır. Üç adet mantıksal operatör vardır: and (ve), or (veya) ve not (değil). Bu basit Boolean deyimlerini kullanarak, daha karmaşık Boolean ifadeleri üretebiliriz. Bu operatörlerin anlamları, parantez içinde yazılmış olan Türkçe anlamlarıyla benzerdir. Örneğin, x > 0 ve x < 10 ifadesi, "x ifadesi 0'dan büyük ve aynı zamanda x 10'dan küçük olduğunda" doğrudur.

```
x = int(input("Lütfen bir sayı giriniz: "))

if x > 0 and x < 10:
    print("x sayısı 0 ile 10 arasındadır.")
elif x < 0 or x > 10:
    print("x sayısı 10 dan büyük VEYA 0 dan küçüktür.")
```

```
Lütfen bir sayı giriniz: 0
```

```
yas = int(input("Lütfen bir sayı giriniz: "))

if yas <= 0:
```

```
    print("Henüz doğmadınız")
elif yas > 0 and yas < 18:
    print("Üzgünüm, henüz ehliyet alamazsınız")
else:
    print("Ehliyet alabilirsiniz!")
```

```
Lütfen bir sayı giriniz: 98
Ehliyet alabilirsiniz!
```

Daha karmaşık ifadeler de yazabilir ve önceliklerinize göre parantezlere alabilirsiniz.

```
sayi = int(input("Lütfen bir sayı giriniz: "))
istenmeyen_sayi = int(input("Lütfen istenmeyen bir sayı giriniz: "))

if (x > 0 and x < 10) and not x == istenmeyen_sayi:
    print("x, 0 ile 10 arasındadır ve", istenmeyen_sayi, "ile x eşit değildir.")
```

```
Lütfen bir sayı giriniz: 5
Lütfen istenmeyen bir sayı giriniz: 6
x, 0 ile 10 arasındadır ve 6 ile x eşit değildir.
```

Aşağıda mantıksal operatörlerin doğruluk tabloları verilmiştir.

7.0.1 or Operatörü

1. İfade	2. İfade	1. İfade OR 2. İfade
True	True	True
True	False	True
False	True	True
False	False	False

7.0.2 and Operatörü

1. İfade	2. İfade	1. İfade AND 2. İfade
True	True	True
True	False	False
False	True	False
False	False	False

UYARI: not operatörü önüne geldiği her şeyin doğruluk değerini tersine çevirmekle görevlidir.

```
not True
```

```
False
```

```
not 0
```

```
True
```

Matematiksel Operatörler

Python'da kullanılabilen matematiksel operatörler ve karşılıkları aşağıdaki tabloda verilmiştir.

Operatör	Açıklama	Syntax
+	Toplama işareti	x + y
-	Çıkarma işareti	x - y
*	Çarpma işareti	x * y
/	Bölme işareti	x / y
//	İnteger bölmesi işareti, bölmenin tamsayı kısmını döndürür	x // y
%	Mod işareti, birincinin ikinciye bölümünden kalanı verir	x % y
**	Kuvvet/Üs işareti, birinci sayı ^ ikinci sayı	x ** y

```python
# Matematiksel Operatör Örnekleri
a = 9
b = 4

ekle = a + b # Toplama işlemi

cikar = a - b # Çıkarma işlemi

carp = a * b # Çarpma İşlemi

bol1 = a / b # Bölme işlemi

bol2 = a // b # İnteger bölmesi işlemi

mod = a % b # mod alma işlemi

ussu = a ** b

print(ekle)
print(cikar)
print(carp)
print(bol1)
print(bol2)
print(mod)
print(ussu)
```

```
13
5
36
2.25
2
1
6561
```