

KONU 7

Koşul İfadeleri

Kazanımlar :

- Koşul Nedir?
 - Koşullar Ne İçin Kullanılır?
 - Koşul İfadeleri
-

Koşul Nedir?

Koşul: Bir yargının ya da düşüncenin gerçekleşmesinin diğerine bağlı olduğu cümlelerdir

Örnek: Sınavdan 70 ve üzeri alırsa geçer.

Koşullar Ne İçin Kullanılır?

Koşullar yazılımda çok önemli yer tutarlar.
Tüm programların çalışması koşullardan
ibarettir.

Koşullar yerine geldiğinde bilgisayara
herhangi bir işlem yapmasını söyleyebiliriz.

Koşul ifadeleri

- **İf (ise)**
- **elif (*ise)**
- **else (yoksa/aksi halde)**

Koşul ifadeleri

```
if <koşul>:  
    <Komut Bloğu>
```

```
elif <koşul>:  
    <Komut Bloğu>
```

```
else:  
    <Komut Bloğu>
```

İf (ise)

```
x = True  
if x:  
    print("Merhaba")
```

DİKKAT!!!



İf içindeki ifadeler

1 TAB boşluğu içerisinde

Koşul ifadeleri Boolean değişkeni ile kullanılır. True olduğunda bloğun içine girer.

İf (ise)

Karşılaştırma ve Mantıksal Operatörleri
HATIRLAYIN!!!!

```
x = 100
if x == 100:
    print("x = 100")
```

DİKKAT!!!

İf içindeki ifadeler

1 TAB boşluğu içerisinde

İf (ise)

```
x = 6  
if x == 100:  
    print("x = 100")
```

DİKKAT!!!

İf içindeki ifadeler
1 TAB boşluğu içerisinde

DİKKAT!!!

İf bloğunun dışında

if (ise)

```
x = 50
if x != 100 and x > 20:
    print(x)

print("Deneme")
```



DİKKAT!!!

İf içindeki ifadeler

1 TAB boşluğu içerisinde

if (ise)

```
sinav1 = 30
```

```
sinav2 = 70
```

```
ortalama = (sinav1 + sinav2)/2
```

```
if ortalama > 50:
```

```
    print("Dersi Geçtin.")
```



DİKKAT!!!

If içindeki ifadeler

1 TAB boşluğu içerisinde

if (ise)

```
sinav1 = 30
```

```
sinav2 = 70
```

```
ortalama = (sinav1 + sinav2)/2
```

```
if ortalama > 45 and sinav2 > 45:  
    print("Dersi Geçtin.")
```



DİKKAT!!!

İf içindeki ifadeler

1 TAB boşluğu içerisinde

elif (ise)

elif (Else if) yapısı **if** veya **elif** 'ten sonra kullanılır. Tek başına kullanılamaz. Eğer üstteki koşullar sağlanmadıysa **elif** koşulunu dener.

elif (ise)

```
x = False
if x:
    print("Merhaba")
elif not x:
    print("Hoşçakal")
```

elif (ise)

```
x = False
if x:
    print("Merhaba")
if not x:
    print("Hoşçakal")
```

elif (ise)

QUIZ:

Az önceki örnekler arasındaki fark nedir?

```
x = False
if x:
    print("Merhaba")
elif not x:
    print("Hoşçakal")
```

```
x = False
if x:
    print("Merhaba")
if not x:
    print("Hoşçakal")
```

elif (ise)

CEVAP: IF-ELIF bloklarında herhangi bir koşula girildikten sonra programa kaldığı yerden devam eder. Diğer Koşulları denemez.

```
x = 98  
if x > 95:  
    print("A1")  
elif x > 90:  
    print("A2")
```

```
x = 98  
if x > 95:  
    print("A1")  
if x > 90:  
    print("A2")
```

else (yoksa/aksi halde)

else bloğu **if** veya **if-elif** bloklarının en sonunda kullanılır. Hiç bir koşul sağlanmamışsa bu koşula girer.

****else** herhangi bir koşulu kontrol etmez.**

NOT: Koşul blokları iç içe kullanılabilir!

else (yoksa/aksi halde)

```
x = int(input("Notunuz: " ))
```

```
if x >= 0 and x <= 100:
```

...devam edecek...

```
else:
```

```
    print("Geçersiz Not")
```

else (yoksa/aksi halde)

```
x = int(input("Notunuz: "))

if x >= 0 and x <= 100:

    if x >= 95:
        print("A1")

    elif x >= 90: ← Neden
        print("A2")      x >= 90 and x < 95
    elif x >= 85:
        print("A3")      yazmadık?

    else:
        print("B")

else:
    print("Geçersiz Not")
```

NOT ! !

```
x = int(input("Notunuz: "))

if x >= 0 and x <= 100:

    if x >= 95:
        print("A1")

    if x >= 90:
        print("A2")

    elif x >= 85:
        print("A3")

    else:
        print("B")

else:
    print("Geçersiz Not")
```

Your Score 85%

You FAILEDPASSED the
Exam

Required Score 85%

@coderhumor

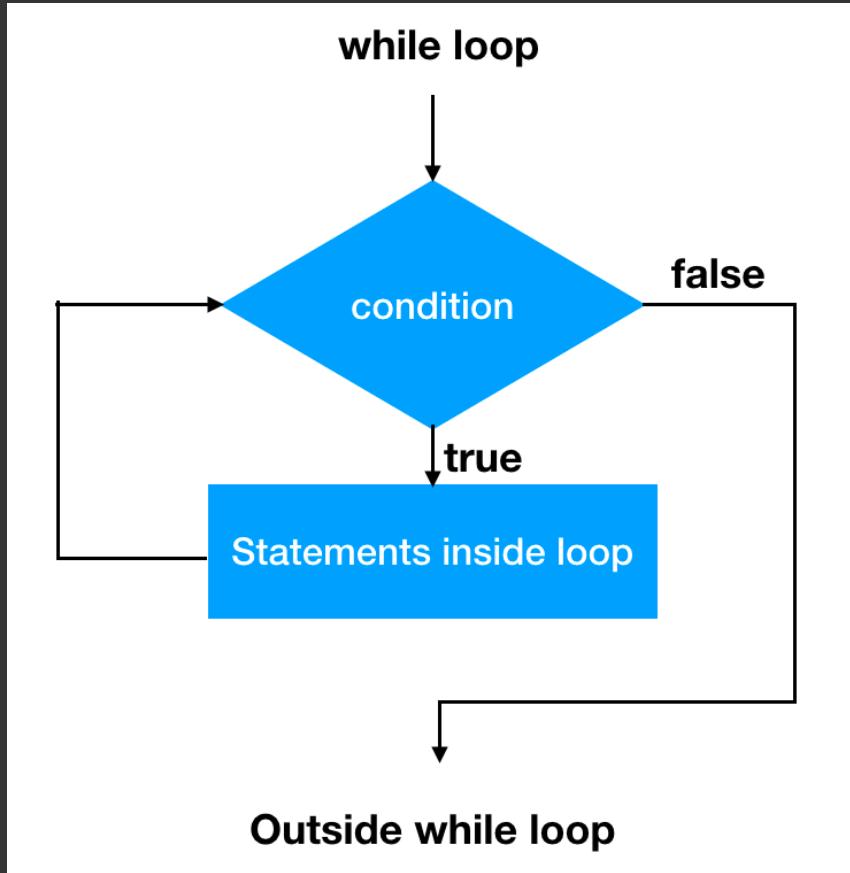
KONU 8

While Döngüsü

Kazanımlar :

- Döngü Nedir?
 - While Döngüsü Nedir?
 - En Çok Kullanıldığı Yer?
-

Döngü (Loop) Nedir?



– Döngü (Loop) Nedir?

Döngüler Boolean değişkeni ile kullanılır.
True olduğunda program içinde döner.

While Döngüsü

```
while <koşul>:  
    <Komut Bloğu>
```

While Döngüsü

```
a = 1  
while a <= 5:  
    print(a)          Sonuç nedir?
```

SONSUZ DÖNGÜ

While Döngüsü

```
a = 1  
while a <= 5:          Sonuç nedir?  
    print(a)  
    a += 1
```

Sonsuz While Döngüleri

Programlamada en çok kullanılan döngü çeşididir. Kullandığınız her şey (**Gerçekten HER ŞEY**) bu döngü sayesinde vardır.

Örnek: İşletim sistemleri, Oyunlar, Telefonlar ve aklınıza ne geliyorsa.

Sonsuz While Döngüleri

```
while True:  
    print("A")
```

```
while 1:  
    print("A")
```



0 dışındaki tüm sayılar
True olarak değerlendirilir.

Sonsuz While Döngüleri

Peki...

While döngüsünden çıkışın gerekliliği?

Sonsuz bir döngüden nasıl çıkış yapılabiliriz?

Let's break.

Sonsuz While Döngüleri

break : break kelimesi kullanıldığı zaman
 break (Döngüyü Bitir)
kelimesinin bulunduğu ilk döngüden çıkar.

Sonsuz While Döngüleri

```
while True:  
    parola = input("Parola: ")  
    if len(parola) >= 8:  
        print("parola belirlendi.")  
        break  
    else:  
        print("Parolanız kısa.")
```

Sonsuz While Döngüleri

```
a = 0
while a < 2:
    while True:
        parola = input("Parola: ")
        if len(parola) >= 8:
            print("parola belirlendi.")
            break
        else:
            print("Parolanız kısa.")
    a += 1
```

Sonsuz While Döngüleri

Ayrıca bilmemiz gereken bir de **continue** (Döngüye baştan devam et) kelimesi bulunmakta.

Try-except yapısında ayrıntılı bahsedilecektir

KONU 9

For Döngüsü

Kazanımlar :

- For Döngüsü Nedir?
 - En Çok Kullanıldığı Yer?
-

For Döngüsü

Programlamada sıkça kullanılır.

Örnek: Kullanıcı listesinin gösterilmesi

Python'da iki çeşit önemli
kullanımı vardır.

For Döngüsü

```
for <değişken> in <liste>:
```

<Komut Bloğu>

```
for <değişken> in range(<değişken>):
```

<Komut Bloğu>

For Döngüsü

```
for <değişken> in <liste>:  
    <Komut Bloğu>
```

Listeleri şuan görmemeye rağmen,
Bu for döngüsü çeşidi <liste>'nin
elemanlarını teker teker <değişken>e atar.

Unutmayın STRING'lerde bir listedir!!

For Döngüsü

```
turkce_karakterler = "şçöğü"
```

```
parola = input("Parolanız: ")
```

DİKKAT!!!

İki in'in anlamı farklı!

```
for karakterler in parola:  
    if karakterler in turkce_karakterler:  
        print("Türkçe karakter kullanma!")
```

Şifrenin içinde Ç,Ş,Ö,Ğ,Ü olsaydı ne olacaktı?

For Döngüsü

```
turkce_karakterler = "şçöğüı"
```

```
parola = input("Parolanız: ")
```

```
for karakterler in parola:
```

```
    if karakterler.lower() in turkce_karakterler:  
        print("Türkçe karakter kullanma!")
```

Şifrede gelen büyük harfleri
küçük harfe çevirerek kontrol
ediyoruz

For Döngüsü

```
for <değişken> in range(<değişken>):  
    <Komut Bloğu>
```

for döngüsünün `range()` fonksiyonuyla
birlikte kullanımı çok yaygındır.

Bu döngüyü incelemeden önce
`range()` fonksiyonunu inceleyeceğiz.

- **range()** fonksiyonu

```
sonuc1 = range(10) #0' dan 10 a kadar  
sonuc2 = range(2,10) #2' dan 10 a kadar  
sonuc3 = range(2,10,2) #2' dan 10 a kadar 2 şer
```

```
sonuc1 = [0,1,2,3,4,5,6,7,8,9]  
sonuc2 = [2,3,4,5,6,7,8,9]  
sonuc3 = [2,4,6,8]
```

range() fonksiyonu verilen parametrelerle
göre bir liste döndürür bu yüzden
for döngüsü aslında yine listelerle
çalışmış olur.

- range() fonksiyonu

```
for i in range(5):  
    print(i)  
  
for i in range(-2, 12):  
    print(i)  
  
for i in range(5, 8, 2):  
    print(i)
```

—

Örnek 1 'e başlıyoruz.

KONU 10

Fonksiyonlar

Kazanımlar :

- Fonksiyon Nedir?
 - Fonksiyon Tipleri?
 - Python'da fonksiyon tanımlama
-

Fonksiyon Nedir?

Yazılım dillerinin olmazsa olmazlarıdır.

Belli bir amaç için yazılmış kodlar
bütünüdür.

Okunurluğu arttırrır.

Hataları azaltır.

Bir çok kez çağrırlacak kod satırlarını
tek satırda çağrırbilmemize imkan tanır.

Fonksiyon Tipleri?

Fonksiyonlar iki çeşide ayrılırlar.

- 1) Değer döndürenler
- 2) Değer döndürmeyenler

Bunlar için örnekleri inceleyeceğiz.

Bildiğimiz Fonksiyonlar

`print()`

`input()`

Fonksiyonların Tiplerini

(Değişken Döndüren veya Döndürmeyen) ,

Kullandığımız bu iki fonksiyonu anlayarak
kavrayabiliriz.

Fonksiyon Tanımlama

```
def fonksiyonismi(parametreler) :  
    <Fonksiyon içinde yapılacaklar>
```

Fonksiyon Tanımlama

```
def print(parametreler):  
    <Fonksiyon içinde yapılacaklar>
```

```
def input(parametreler):  
    <Fonksiyon içinde yapılacaklar>  
    return veri
```

Fonksiyon Tanımlama

```
name = input("İsminiz : ")  
# Dönen bir değer olduğu için değişkene atıyoruz  
print(name)  
  
name = input()  
  
print(name)  
# Dönen bir değer olmadığı için değişkene atamıyoruz
```

Fonksiyon Tanımlama

```
def topla(x,y):  
    sonuc = x+y  
    return sonuc
```

```
def mesaj():  
    print("DENEME")
```

```
def parola():  
    return "12345"
```

```
def merhaba(name):  
    print("Merhaba",name)
```

```
sonuc2 = topla(2,1)  
print(sonuc)
```

```
mesaj()
```

```
print(parola())
```

```
merhaba("Kaan")
```

HATA

Fonksiyon içindeki değişkenler
Fonksiyon dışarısında kullanılmaz.

- Global ve Local (Yerel) Değişken

Fonksiyonların içindeki değişkenlerin dışında kullanılamadığını öğrendik.

Peki Neden?

Fonksiyonların içindeki değişkenler Local değişken olarak adlandırılır.

En dışarıdaki değişkenlerimizde Python' da global değişkenlerdir.

- Global ve Local (Yerel) Değişken

Peki **global** Değişkenler fonksiyon içinde değiştirilebilir mi?

Evet. Ancak bunun için fonksiyonun içinde değişkenin **global** olduğunu belirtmek koşulu ile.

- Global ve Local (Yerel) Değişken

Parametrelerde z olmadığı için
global olduğunu anladı.

`z = 10`

`def f(x):`

`return z+x` Burada gerçekten
global olduğunu belirttik.

`print(f(5))`
`print(z)`

`z = 10`

`def f(x):`
 `global z`
 `z = z+x`
 `return z`

Bu sayede fonksiyonun içinden
değişkeni değiştirebildik `print(f(5))`
`print(z)`

Fonksiyon Tanımlama

Fonksiyonlar birden fazla değer döndürebilir

```
def islem(x,y):  
    toplam = x+y  
    cikarma = x-y  
    return toplam,cikarma
```

```
t,c = islem(10,5)
```



Aslında tek değişken gönderiyor.
ARAŞTIR: tuple

KONU 11

Kütüphaneler (Modül)

Kazanımlar :

- Kütüphane Nedir?
 - Neden Varlar?
 - Nasıl Kütüphane Oluşturulur?
-

Kütüphane Nedir?

Kütüphane bir konu üzerinde daha önceden hazırlanmış ve çalıştığı onaylanmış fonksiyonların derlenmesidir.

Bazı önemli kütüphaneler:
OpenCV, Numpy, TensorFlow

Kütüphaneler Neden Kullanılır?

QUIZ:

Sorunun cevabı çok zor değil.

Nasıl Kütüphane Oluşturulur?

Basit olarak fonksiyonlarınızı bir .py uzantılı bir dosyaya yazıp istediğiniz isimde kaydetmeniz yeterlidir.

Artık bir kütüphanemiz var.

Gelin canlı olarak bir kaç adet kütüphaneyi inceleyelim.

(this,operator)

KONU 12

Kütüphane ve Fonksiyon
Ekleme

Kazanımlar :

- Kütüphane Ekleme
 - Fonksiyon Ekleme
-

Kütüphane Ekleme

Kütüphane oluşturmayı öğrendik.

Şimdi sıra kodumuza hazır bir kütüphane eklemekte.

Bunun için sihirli kelimeler
import ve **from** — **import**

```
1 import sys
2 sys.path.insert(1, './assests/')
3 from common_funcs import print_ex1
4 #Bu satırın üstünü değiştirmeyin.
5
6 x = 2
7 y = 10
8 string = "Kaan ile Python Öğreniyorum"
9 ondalik_sayim = 52.2
10
11 #Değişkenler bu satırın üstünde tanımlanacak.
12 print("====")
13 print("Değişkenler tanımlandığındaki değerler.")
14 print("====")
15 print_ex1(x,y,string,ondalik_sayim)# Değerleri bastıran bir fonks.
16 #Değişkenler bu satırın altında değiştirilecek.
17
18 x = 22
19 y = 20
20 string = "asdasd ile Python Öğreniyorum"
21 ondalik_sayim = 3.3
```

Hafta 1
Örnek 3

Kütüphane Ekleme

```
import <kütüphane adı>
```

Kütüphane içindeki değişkenleri ve fonksiyonları kullanma:

```
<kutuphane adı>.<fonksiyon adı>()  
<kutuphane adı>.<değişken adı>
```

Kütüphane Ekleme

```
import os  
print(os.name)  
print(os.getcwd())
```

OS kütüphanesindeki
bir değişken

OS kütüphanesindeki
bir fonksiyon

Kütüphane Ekleme

```
from <kütüphane adı> import <eklenecekler>
```

Kütüphane içindeki değişkenleri ve fonksiyonları kullanma:

Doğrudan ekleme

```
<fonksiyon adı>()
```

```
<değişken adı>
```

Kütüphane Ekleme

```
from os import name, getcwd  
print(name)  
print(getcwd())
```

OS kütüphanesindeki
bir değişken

Artık fonksiyonlar ve değişkenler
doğrudan kullanılabilir.

OS kütüphanesindeki
bir fonksiyon

—

Örnek 2 'e başlıyoruz.

KONU 13

Listeler

Kazanımlar :

- Liste Nedir?
 - Nerelerde Kullanılır?
-

Listeler

Listeler yazılımlarda çok kullanılırlar.

Python'da aynı veya farklı değişken tiplerinden çok sayıda yanyana tutmaya olanak sağlar

Örnek: STRING'ler karakterlerin bulunduğu bir listedir.

Listeler

INDEXLER ?

Start:End with Indexes to print Range

Slicing from here till end

0 1 2 3 4 5 6 7 8 9 10 11 12 13

G E E K S F O R G E E K S

-13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

[:

Default Beginning of Sequence

Slicing from here till beginning

Reverse String by using [: :-1]

:]

Default End of Sequence

- Liste Tanımlama

```
liste = [] # boş liste tanımlama
```

```
liste = ["Text",True,1920,5.2]
```

- Listedede İstenilen Elemana Ulaşmak

0 1 2 3

0 1 2 3

```
liste = ["Text",True,1920,5.2]
```

```
name = "Kaan"
```

```
print(name[2])
```

```
print(liste[1])
```

```
print(liste[0][3])
```

Listede Yeni Eleman Ekleme

Bu işlem için `append()` metodu
kullanılır.

```
liste.append(<eklemekistenen>)
```

Listede Yeni Eleman Ekleme

```
liste = ["Text",True,1920,5.2]
```

```
name = "Kaan"
```

```
x = 1050
```

```
liste.append(name)
```

```
liste.append(x)
```

```
print(liste)
```

- Listedede Yeni Eleman Ekleme (**)

`append()` metodunu gördük.

Ama listenin sonuna değil istediğimiz bir yere eklememiz gerekiyorsa ne yapacağız?

```
liste.insert(<index no>,<eklemekistenen>)
```

- Listedede Yeni Eleman Ekleme (**)

```
liste = ["Text",True,1920,5.2]
```

```
name = "Kaan"
```

```
x = 1050
```

```
liste.insert(1,name)
```

```
liste.insert(0,x)
```

```
print(liste)
```

Listeden Eleman Silme

`liste.remove(<eklemek istenilen>`

`veya`

`del liste[<index>]`

- Listedede Yeni Eleman Ekleme (**)

```
liste = ["Text",True,1920,5.2]
```

```
name = "Text"
```

```
x = 1050
```

```
liste.remove(name)
```

```
del liste[1]
```

```
print(liste)
```

Listeleri Kopyalama

```
liste = liste2
```

Kopyalamak değildir!

Sadece aynı değişkeni farklı bir isimlede kullanmaya yarar. (Reference)

Doğrusu: `copy()` methodunu kullanmaktadır.

```
liste = liste2.copy()
```

- Listedede Yeni Eleman Ekleme (**)

```
liste2 = ["Text",True,1920,5.2]
```

```
list1 = liste2
```

```
list2 = liste2.copy()
```

```
del liste2[0]
```

```
print(list1)
```

```
print(list2)
```

—

Örnek 3 'e başlıyoruz.