PYTHON 101

e-bursum

3. GÜN
BENGÜ YURDAKUL

10 Mart 2021

MANTIKSAL OPERATÖRLER

- Python' da mantıksal operatörleri birden fazla koşulu/ifadeyi birlikte değerlendirmek, birbirleriyle karşılaştırmak için kullanırız. Genelde bir koşullu ifadenin yanında kullanılır.

- Örnek vermek gerekirse dondurma alıp almadığımızı kontrol ettiğimiz bir durum olsun. Bunun için en az iki şeyi kontrol etmemiz gerekiyor : dondurmacı dondurmayı hazırladı mı VE biz bu dondurmayı elimize aldık mı ? Görüldüğü üzere aradaki VE bizim için bir mantıksal operatördür.

VE (AND)

→ Ve operatörü and şeklinde yazılır.

Burada kontrol ettiğimiz ifadelerin ikisi de (veya hepsi) doğru ise sonuç doğru (`True`) olur, en az birinin yanlış olma durumunda sonuç da yanlış (`False`) olacaktır.

Örnek vermemiz gerekirse komut satırına şu ifadeyi yazalım :

```
print(8 < 10) and print(6 > 5)
```

İki tarafın da doğru olduğunu biliyoruz, hem 10, 8'den büyüktür hem de 6, 5'ten büyüktür o halde çalıştırdığımızda Python'ın bize `True` çıktısını verecektir.

VE (AND)

→ Ve operatörü and şeklinde yazılır.

Burada kontrol ettiğimiz ifadelerin ikisi de (veya hepsi) doğru ise sonuç doğru (`True`) olur, en az birinin yanlış olma durumunda sonuç da yanlış (`False`) olacaktır.

Başka bir örnek daha yapalım, peki şu şekilde yazsaydık sonuç ne olurdu ?

```
print(8 > 10) and print(6 > 5)
```

Göreceğiniz üzere bu sefer bir taraf yanlış olduğu için Python bize `False` çıktısını verecektir.

VEYA (OR)

→ Veya operatörü or şeklinde yazılır.

Bu sefer doğru (`True`) çıktısını almamız için en az bir tarafın doğru olması yeterlidir. Yani hem iki tarafın doğru olma durumunda hem de içlerinden birisinin doğru olması durumunda Python bize `True` çıktısını verecektir.

Örnek vermemiz gerekirse komut satırına şu ifadeyi yazalım :

```
print(8 == 10) or print(8 == 8)
```

Bu ifadeyi incelersek eğer, sol taraftaki (8==10) durum yanlış iken sağ taraftaki (8==8) durum doğrudur. Ne demiştik, taraflardan en az birinin doğru olması durumunda çıktımız doğru (`True`) olacaktır.

Peki or bize ne zaman `False` değerini döndürecektir ? İfadelerin her ikisinin de yanlış olması durumunda. Mesela aşağıdaki örneği yazarak Python'ın `False` çıktısı verdiğini görebilirsiniz.

```
print(8 > 10) or print(8 > 8)
```

DEĞİL (NOT)

→ Son mantıksal operatörümüz ise not şeklinde yazılan ‘değil’

Kısacası yazdığımız ifadenin başına not yazarak o ifadenin sonucunu tersine çeviriyoruz.True ise `False`, False ise `True` olur.

Aşağıdaki örneklerle daha ne anlayabiliriz.

```
print(not(8 == 8))
```

8, 8’e eşit olduğu için ifademizin sonucu `True` oldu. Ancak başında değil operatörümüz olduğu için bu sonucu tersine çeviriyoruz ve programdan `False` çıktısını alıyoruz.

Aynı şekilde ;

```
print(not(1>2))
```

yanlış olan bu ifademiz de başındaki not sebebiyle tersi olan `True` sonucunu bize verecektir.

if KOŞULLU DURUMU

- Program yazarken bazı koşullara bağlı olarak gerçekleşmesini istediğimiz komutlara/durumlara ihtiyacımız olabilir. Mesela çift sayıları ekrana basacak bir programın, sayının çift olması durumunda o sayıyı basması gerekiyor, eğer tek sayı ise bu işlem/komut gerçekleşmemelidir. Bu tarz durumlarda yardımıza if koşullu ifadeleri koşar.

- If bir çeşit koşul durumudur. Yanında yazan şart/ifade sağlanmadıkça program istenen komutu gerçekleştirmez.

IF

Yazılışı şu şekildedir :

```
if koşul :

    koşul sağlanıyorsa gerçekleşecek işlemler
```

İsteğe bağlı olarak koşul ifadesi parantez içine alınabilir.

```
if (koşul) :

    koşul sağlanıyorsa gerçekleşecek işlemler
```

if koşullu durumunun altına istendiği kadar işlem yazılabilir, ancak girintiye dikkat edilmelidir. Girinti yapmayı bıraktığınız anda program if koşullu durumundan çıkacaktır.

IF

Örnek vermemiz gerekirse komut satırına şu ifadeyi yazalım :

```
a = 10

b = 8

if a > b :

    print(a*b)
```

Yazdığımız kodu açıklamam gerekirse önce a değişkenine 10 sayısını ve b değişkenine 8 sayısını atadık. Ardından if koşullu durumunu yazarak yanında sağlanmasını istediğimiz koşulu belirttik. Yani a‘nın b’den büyük olması durumunda bir içerideki işlem gerçekleşecektir. 10, 8’den büyük olduğu için koşulumuz sağlandı ve içerideki işleme geldik. Burada da 10 ile 8’i çarparak ekrana bastık.

IF

Ancak b yerine 15 yazsaydık if ‘in yanındaki koşul sağlanmayacağı için program içeriye girmeyecek ve ekrana hiçbir şey bastırmayacaktı.Yazdığımız kodu açıklamam gerekirse önce a değişkenine 10 sayısını ve b değişkenine 8 sayısını atadık. Ardından if koşullu durumunu yazarak yanında sağlanmasını istediğimiz koşulu belirttik. Yani a‘nın b’den büyük olması durumunda bir içerideki işlem gerçekleşecektir. 10, 8’den büyük olduğu için koşulumuz sağlandı ve içerideki işleme geldik. Burada da 10 ile 8’i çarparak ekrana bastık.

Ancak b yerine 15 yazsaydık if ‘in yanındaki koşul sağlanmayacağı için program içeriye girmeyecek ve ekrana hiçbir şey bastırmayacaktı.

ELSE

- Program yazarken birden fazla koşulu kontrol etmek veya if durumu sağlanırsa başka işlemler, sağlanmazsa başka işlemler yaptırmak istediğimizde if ile birlikte else durumuna da ihtiyacımız olacaktır.

- Else kısaca ‘if koşulu sağlanmıyorsa’ durumudur.

Yazılışı şu şekildedir :

```
if koşul :
    koşul sağlanıyorsa gerçekleşecek işlemler
else :
    if’in yanına yazdığımız koşul sağlanmıyorsa gerçekleşecek işlemler
```

ELSE

Mantığı daha iyi kavramak için hızlıca bir örneğe bakalım.

```
a = 10

b = 8

if a < b :

    print(a*b)

else :

    print(a/b)
```

İlk olarak a değişkenine 10, b değişkenine ise 8 sayısını atadık. Ardından if koşullu durumunu devreye soktuk, yanında da şartımızı (b, a'dan büyüktür.) yazdık. Eğer bu şart sağlanırsa program, if içindeki işlemleri yapacak, sağlanmazsa else'in içindeki işlemleri yapacak.

ELSE

Ancak 8, 10'dan büyük olmadığı için if'in şartı sağlanamadı ve else'in (if şartı sağlanmıyorsa durumuydu, hatırlayınız) içindeki işlemi yaptı.

Bir örneğe daha bakalım :

```
Sayi = -5

if sayi > 0 :

    print(sayi, "pozitif bir sayıdır.")

else :

    print(sayi, "negatif bir sayıdır.")
```

→ if tek başına kullanılabilir ancak else kullanabilmek için mutlaka bir if 'e ihtiyacımız var. Çünkü bir if durumunun değil halidir.

ELIF

- Birden fazla koşulun olduğu ve bu koşulların kontrol edilmesi gerektiği durumlarda elif yardımımıza koşar. Yukarıda görmüş olduğumuz else'in, yanında koşulu olan halidir diye düşünebilirsiniz.

- Çoğu programlama dilinde else if şeklinde kullanılırken Python dilinde elif olarak kullanılmaktadır. Aynı else'de olduğu gibi if olmadan tek başına elif (else if) kullanamayız çünkü başında bir else vardır ve her else bir if'e ihtiyaç duyar.

- else, if durumu sağlanmıyorsa demek olduğu için else yazdıktan sonra tekrar koşul belirtmeye gerek yok. Ancak elif (else if) 'bir önceki if koşulu sağlanmıyor şimdi de bu koşul sağlanıyor mu bakalım' durumları için kullanılır ve if ile bittiği için koşul belirtilir.

ELIF

Yazılışı şu şekildedir :

```
if koşul:
    koşul doğru ise yapılacak işlemler
elif koşul:
    koşul doğru ise yapılacak işlemler
else:
    koşul yanlış ise yapılacak işlemler
```

Buradaki olayımız şu : Önce if'in yanındaki koşula bakıyoruz eğer bu koşul sağlanıyorsa program içindeki işlemi yapıyor. Ancak bu koşul sağlanmıyorsa hemen altındaki elif ifadesine geliyoruz ve buradaki koşulu kontrol ediyoruz. Bu koşul sağlanıyorsa elif'in içindeki işlemler yapılıyor. Ancak bu koşul da sağlanmıyorsa en alttaki else'in içindeki işlemler yapılıyor.

ELIF

```
sayi = 0

if sayi > 0:

    print(sayi, "pozitif bir sayıdır.")

elif sayi == 0:

    print(sayi, "sayısı sıfıra eşittir.")

else:

    print(sayi, "negatif bir sayıdır.")
```

sayi değişkenine 0 değerini atadıktan sonra ilk koşulumuzu kontrol ediyoruz : 0, 0'dan büyük olmadığı için elif durumuna geçip oradaki koşulumuzu kontrol ediyoruz. 0, 0'a eşit olduğu için elif'in altındaki işlemi gerçekleştiriyoruz.

→ Program yazarken istediğimiz kadar elif kullanabiliyoruz, herhangi bir sınırlandırma yok.

GİRDİ (INPUT) ALMA

- Bu noktada artık her şeyi kendimiz yazmayı bırakıyor ve kullanıcıdan veri istemeye başlıyoruz. Input yardımıyla klavyeden / kullanıcıdan bilgi/veri isteyip üstünde işlemler yapacağız.

- Programlama dillerinin olmazsa olmazı olan input fonksiyonu Python'da input( ) şeklinde yazılır ve her zaman aldığı veriyi string tipinden alır.

input( )

- Bu nedenle sayısal işlemler yaparken dönüştürmeye ve dönüştüreceğimiz veri tipine dikkat etmeliyiz.
- Burada kullanıcıdan isteklerimiz doğrultusunda bir girdi yapmasını isteyeceğiz ve o girdiyi bir değişkene atayıp kullanacağız.

`a = input( )` dediğimizde program, kullanıcıdan bir veri girmesini isteyecektir ve girilen veriyi a değişkenine atayacaktır.

Basit bir örnekle deneyelim :

```
a = input()

print(“Merhaba”, a)
```

Çalıştırdıktan sonra ekrana ne yazarsak program ekrana Merhaba yazısının yanında yazdığımız şeyi basacaktır. Mesela girdi olarak Dünya yazsaydım ekranda `Merhaba Dünya` yazısı çıkardı.

input( )

Aynı şekilde şu kodu da deneyebiliriz :

```
a = input(“İsminizi giriniz : “)

print(“Merhaba”, a)
```

Bu örneği kendi komut satırımıza yazalım ve çalıştıralım. Ekranda

`İsminizi giriniz :`

yazısının çıktığını göreceğiz. Buraya isimimizi girdikten sonra program bize `Merhaba <isim>` çıktısını verecektir. Yani Bengü yazmış olsaydım çıktı `Merhaba Bengü` şeklinde olacaktı.

Yani input( ) fonksiyonunun içine tırnak işareti ile (“ “) bir şey yazdığınızda program bunu olduğu basacak ve sonunda sizden bir girdi isteyecektir. Ardından da kodunuzun devamına göre işlemleri yapacaktır.

input( ) TİP DÖNÜŞÜMLERİ

Yukarıda verdiğim örnekte input istendiğinde bir sayı girdiyseniz eğer işlemleri yapamadığını ve hata verdiğini görmüşsünüzdür. Bunun sebebi en başta belirttiğim gibi bu fonksiyonun her şeyi string (karakter dizisi) olarak algılaması. Bu nedenden dolayı başka veri tipine ihtiyacımız olduğunda tip dönüşümü yapmamız gerekir.

Tip dönüşümleri için yapmamız gereken şey dönüştürmek istediğimiz veri tipini yazıp parantez içinde input fonksiyonunu yazmak.

```
int(input( ))
```

```
float(input( ))
```

gibi.

Kullanıcıdan alınan string tipindeki veri tipini istenen(başına yazılan) veri tipine dönüştürecektir, eğer başına int yazarsak alınan veri int tipinde olacaktır.

input( ) Tip Dönüşümleri

```
a = int(input( ))

b = int(input( ))

print(a * b)
```

kodunu çalıştıralım. Kullanıcıdan iki defa veri girişi isteyecektir. İlk istediği girdiyi integer(int) tipine dönüştürerek a değişkenine, ikinci istediğini de integer tipine dönüştürerek b değişkenine atayacaktır. Ve ardından a ile b 'nin içindeki değerleri çarpacak, print( ) fonksiyonu ile ekrana basacaktır. İlk input isteyişinde 5, ikincisine 6 girersek sonuç `30` olarak çıkacaktır.

Bu kodu sadece input() yazarak deneseydik hata verecekti çünkü girdiğimiz sayıları string(karakter dizisi) tipinde algılayacaktı ve bildiğiniz üzere string veri tiplerinde çarpma işlemi yapılamaz.

input( ) Tip Dönüşümleri

Aynı işlemler float tipinde de geçerlidir.

```
a = float(input(“Dikdörtgenin ilk kenarı : “))

b = float(input((“Dikdörtgenin ikinci kenarı : “ ))

print(“Dikdörtgenin alanı : “, a * b)
```

ilk input’a `4.5` ikincisine de `6.3` girersem `28.349999999999999` sonucunu elde ederim.

input( ) Tip Dönüşümleri

O halde siz de bu kodu komut satırınıza yazarak deneyebilirsiniz :

```
a = int(input("Birinci Sayı:"))

b = int(input("İkinci Sayı:"))

c = int(input("Üçüncü Sayı:"))

print("Toplam:", a+b+c)
```

DÖNGÜLER

- Program yazarken aynı işlemi birden fazla kez tekrarlama ihtiyacı hissedebiliriz.
- Bir koşula bağlı olan tekrarlara döngü denir.
- Python’da for ve while olmak üzere iki farklı temel döngü vardır.

for DÖNGÜSÜ

- Bir çok farklı yazım şekli olan for döngüsünün hangi koşullar altında ve kaç defa döneceğini biz ayarlıyoruz.
- İlk göz atacağımız for döngüsü için bir değişken kullanmamız gerekiyor ve bu değişken genelde « i » harfi oluyor ama siz başka değişkenler de kullanabilirsiniz.

for

Yazımı şu şekilde olabilir ;

```
for i in range (sayıdeğeri) :

    yapılacak işlemler
```

Herhangi bir for döngüsünün yazımına for ile başlanır ve yanına bir değişken konur. Bu değişkenin işlevi, istendiği kadar döngünün dönmesini sağlamaktır. Fark ettiyseniz burada « i » değişkenine bir ilk değer atamadık ve tipini de belirtmedik. Bu noktada Python'ın bize sağladığı bir kolaylıktan faydalanıyoruz. Eğer ilk değeri yoksa Python bu değişkeni 0 olarak düşünecek ve saymaya/dönmeye 0'dan başlayacaktır. Bu yazımda range sırayla gitmek anlamına gelir. Yani sona konan sayıya kadar döngü sırayla dönecektir.

for

```
for i in range (10) :
   print(i)
```

Çalıştırdığımızda programın çıktısı şu olur ;

```
0
1
2
3
4
5
6
7
8
9
```

Öncelikle bir değeri olmadığı için i değişkenimiz 0 kabul ediliyor ve döngü çalışmaya başlıyor. Her turda i'nin değeri bir(1) artacak ve her turda ekrana i'nin değerini basacak. Ancak dikkat edilmesi gereken nokta şu ki i'nin değeri 10'a kadar artacak. Bu durumda 10 sayısı dahil değil.

for

for döngüsündeki değişkenimiz (bu örnekte « i » oluyor) bir sayaç görevi görür.

Ayrıca döngülerin içerisinde yapılacak işlemleri python'un anlayabilmesi için girintilemeye dikkat edilmeli. Eğer herhangi bir girinti olmadan programı yazarsanız döngü hiçbir şey yapmadan sona erecek ve aslında döngünün içinde gerçekleştirilmesini istediğimiz işlemler sadece bir kez yapılacaktır. Yukarıda yazdığımız kodu girintileme olmadan şu şekilde tekrar yazıp çalıştıralım.

for

```
for i in range (10) :

print(i)
```

Bu durumda hata verecek ve girintileme yapmadınız diye sizi uyaracaktır.

Başka bir örnek deneyelim. Bu örnekte de 0'dan başlamadan belli bir sayı aralığında sırayla ilerlemesini isteyelim. O halde yazımı şu şekilde olacaktır ;

```
for i in range (başlangıçsayısı,ekadargideceğisayı) :

    yapılacak işlemler
```

for

başlangıçsayısı kısmında saymaya/dönmeye hangi sayıdan başlamasını istiyorsak onu belirtiyoruz, ekadargideceğisayı kısmında ise hangi sayıya kadar dönmesini/saymasını istediğimizi belirtiyoruz.

Hemen basit bir örneğe göz atalım ;

```
for i in range (4,8) :

    print(i)
```

« i »'nin değeri verilmediği için Python 0 kabul eder demiştik, bu tarz for döngülerinde ise i ilk değer olarak 0'ı değil başlangıçsayısı olarak belirttiğimiz değeri alıyor yani bu örnekte 4'ten başlayacak ve 8'e kadar gidecektir. Ancak yine 8 dahil değildir.

Sonuç olarak alacağımız çıktı şu şekilde olacak ;

```
4

5

6

7
```

Konuyu pekiştirmek adına yeni öğrendiğimiz input() fonksiyonunu kullanarak bir örnek yapalım ;

```
birincisayi = int(input(“Bir sayı giriniz : ”)

ikincisayi = int(input(“Bir sayı giriniz : ”)

for birincisayi in range ikincisayi :

    print(birincisayi)

    print(ikincisayi)
```

for

Örnekleri gördüğümüze göre biraz daha detayına inebiliriz. range fonksiyonunu biraz daha tanıyalım. Türkçede karşılığı “aralık “olan bu fonksiyon sayı aralıklarını belirtir. Kullanımı şu şekildedir;

range(başlangıç değeri , bitiş değeri, değişim miktarı)

range(10) : sıfır ile 9 aralığındaki sayıları belirtir. Başlangıç değerini belirtmediğimiz için varsayılan olarak 0, değişim miktarını belirtmediğimiz için ise varsayılan olarak 1 değerini alır. Ve dikkat etmemiz gereken başka bir husus ise bitiş değerini dahil etmiyor.

for

`range(2,20,3):` 2 ile 19 arasında üçer üçer artarak gider.

```
for i in range (2,20,3) :

   print(i)
```

Alınacak çıktı ;

```
2
5
8
11
14
17
```

for

range(20,2,-2) : 20 ile 2 arasında ikişer ikişer azalarak gider.

```
for i in range (20,2,-2):

   print(i)
```

Alınacak çıktı :

```
20
18
16
14
12
10
8
6
4
```

for

Şimdi ise in ‘e bakalım. Türkçe deki karşılığı “içinde” olan in, istenilen değer ya da aralık içinde olup olmadığını sorgulama yapmamızı sağlayan araçtır.

for i in range (20) : burada sayı değişkeni 0 ile 19 arasındaki(içindeki) bütün değerleri alır.

Sadece integer değerlerle değil aynı zamanda string(karakter dizileri) ile de for döngüsünü kullanabiliyoruz.

Örnek ;

```
for i in “Python” :
    print(i)
```

for

Yine aynı şekilde ilk değeri olmayan i'yi 0 olarak düşünüp yanında yazılan karakter dizisi boyunca sırayla ilerleyerek her harfi (her döngüde) ekrana basacağız. Ancak dikkat edersiniz ki bu sefer sayı aralığı verilmediği için range fonksiyonu kullanılmıyor. Aynı sebepten dolayı bir eksiğine kadar gitme/dönme durumu da söz konusu değildir. Yanda tırnak işareti arasında verilen karakter dizisinin sonuna gelene kadar dönecek ve her aşamada içindeki işlemleri yapacaktır.

Bu durumda çıktımız şu şekilde olacaktır ;

```
P

y

t

h

o

n
```

while DÖNGÜSÜ

- while döngüsünün yazımı ve kullanımı for'a kıyasla daha kolaydır Bu döngü yanında verilen şart sağlandığı sürece döner ve içindeki işlemleri yapar, şart sağlanmıyorsa döngüden çıkar.

- for da olduğu gibi while döngüsünde de değişken(ler) kullanılır ancak bu sefer değişkene bir değer atanmış olmalıdır.

- Python kendisi bu değişkene 0 veya başka değer vermeyecektir. İlk değeri olmayan bir değişken kullanırsanız hatayla karşılaşırsınız. Aynı şekilde for döngüsünde biz yazmadan her turda değişkenin değeri istenen sayı miktarınca artarken while döngüsünde bunu kendimiz ayarlamamız gerekiyor.

while

Yazımı şu şekildedir ;

```
while şartifadesi :
   yapılacak işlemler
```

Bir örnek ile anlatılanları pekiştirelim ;

```
i = 5
while i<10 :
    print(i)
    i = i + 1
```

while

i değişkenine 5 değerini atadık ardından while döngüsünün şartını kontrol ettik. Eğer şart doğru ise döngüye girecek değil ise döngüden çıkacaktır. 5, 10'dan küçük olup şartı sağladığı için döngüye girip ekrana i'nin o anki değerini (yani 5) basıyor. Sonra i'nin değerini bir arttırıp tekrar while'ın başına gelip şartı kontrol ediyor. Bu şekilde şart sağlanmayıncaya kadar dönmeye ve içindeki işlemi yapmaya devam ediyor.

Sonuç olarak çıktımız şu olacaktır ;

5

6

7

8

9

while

Eğer i'nin değerini birer arttırma işlemimizi (i=i+1) yapmamış olsaydık program sonsuz döngüye girerdi. Bu nedenle değişkenin üzerindeki istediğiniz artış/azalış miktarını döngünün içinde işlem olarak yapmamız gerekiyor.

Eğer bunu yapmazsak ne olacağını daha iyi görmek için şu kodu çalıştırabilirsiniz ;

```
sayi = 0

while sayi == 0:

    print("bilgisayar çıldırdı!")
```

while

sayi değeriyle oynadığımız bir işlem yapmadığı için sayi değişkeninin içindeki değer hep 0 olarak kalacak ve değişmeyecektir bundan dolayı döngünün şartı olan sayi==0 şartı sonsuza kadar sağlanacak ve sonsuz döngüye girilecektir. Bunu düzeltmek için kodla azcık oynamamız yeterlidir ;

```
sayi = 0

while sayi == 0 :

    print("bilgisayar çıldırdı!")

    sayi=sayi+1
```

işte problemi düzelttik !

SIRA SİZDE :)

İstediğim ve çalışma örneğini verdiğim kodları yazmanızı bekliyorum. Kendinizi deneyin, denedikten sonra arkadalarda bulunan çözümlere bakabilirsiniz.

Sıra sizde #1

Kullanıcıdan girilen iki sayıdan ilkinin, ikincisinin katı olup olmadığını söyleyen bir program yazınız.

Örnek çıktı ;

```
a sayısı giriniz : 22
b sayısı giriniz : 11
a sayısı b sayısının katıdır.
```

Sıra sizde #2

Kullanıcının girdiği « n » sayısından 1’e kadar olan bütün sayıları toplayıp ekrana yazan program yazınız.

Örnek çıktı ;

```
Bir n sayısı giriniz : 6

20
```

Sıra sizde #3

3 kenarı da girilen üçgenin, eşkenar üçgen olup olmadığını söyleyen program yazınız.

Örnek çıktı ;

```
Birinci kenarı giriniz : 3

İkinci kenarı giriniz : 5

Üçüncü kenarı giriniz : 4

Eşkenar üçgen değildir.
```

Sıra sizde #3 - 2. versiyon

Daha sonra bu programı üçgenin çeşidini de yazacak şekilde geliştiriniz.

Örnek çıktı ;

```
Birinci kenarı giriniz : 4
İkinci kenarı giriniz : 4
Üçüncü kenarı giriniz : 6
Bu bir ikizkenar üçgendir.
```

Sıra sizde #4

Kullanıcıdan 4 farklı sayı girmesini isteyen ve girilen sayıların arasından en büyük olanı ekrana basan program yazınız.

Örnek çıktı ;

```
Bir sayı giriniz : 23

Bir sayı giriniz : 8

Bir sayı giriniz : 39

Bir sayı giriniz : 11

En büyük sayı : 39
```

Sıra sizde #5

Kullanıcıdan iki basamaklı bir sayı girmesini isteyen ve girilen sayının basamaklarının karelerinin toplamını ekrana basan program yazınız.

Örnek çıktı ;

```
İki basamaklı bir sayı giriniz :54

Girdiğiniz sayının basamaklarının kareleri toplamı : 29
```

Sıra sizde #6

Kullanıcıdan en ve boy değerlerini alan ve her satırda sırasıyla *, o, - olacak şekilde dikdörtgen basan program yazınız.

Örnek çıktı ;

```
Enini giriniz : 4

Boyunu giriniz : 5

****

oooo

----

****

Oooo
```

Sıra sizde #1 ÇÖZÜM

```
a = int(input("a sayısı giriniz : "))

b = int(input("b sayısı giriniz : "))


if (a%b==0)

  print("a sayısı b sayısının katıdır. ")

else

  print("a sayısı b sayısının katı değildir. ")
```

Sıra sizde #2 ÇÖZÜM

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

toplam = 0

for n in range (n,1,-1) :

  toplam = toplam + n

print(toplam)
```

Sıra sizde #3 ÇÖZÜM

```
birincikenar = int(input("Birinci kenarı giriniz = "))

ikincikenar = int(input("İkinci kenarı giriniz = "))

ucuncukenar = int(input("Üçüncü kenarı giriniz = "))

if (birincikenar == ikincikenar) :

   if (ikincikenar == ucuncukenar) :

       print("Eşkenar üçgendir.")

   else :

        print("Eşkenar üçgen oluşturmaz.")

else :

        print("Eşkenar üçgen değildir.")
```

Sıra sizde #3-2. versiyon ÇÖZÜM

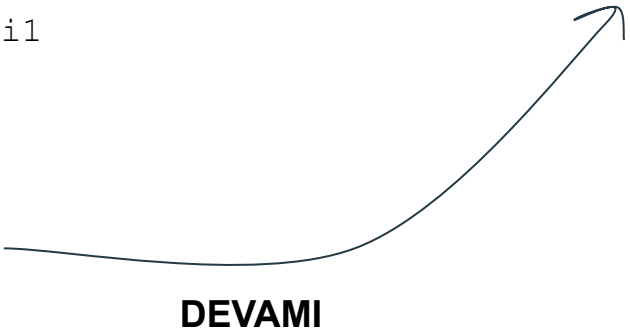
```
birincikenar = int(input("Birinci kenarı giriniz = "))

ikincikenar = int(input("İkinci kenarı giriniz = "))

ucuncukenar = int(input("Üçüncü kenarı giriniz = "))

if (birincikenar == ikincikenar) :

   if (ikincikenar == ucuncukenar) :

       print("Eşkenar üçgendir.")

   else :

        print("İkizkenar üçgendir.")
```

DEVAMI

```
elif (birincikenar == ucuncukenar) :

    print("İkizkenar üçgendir.")

elif (ikincikenar == ucuncukenar) :

    print("ikizkenar üçgendir.")

else :

    print("Çeşitkenar üçgendir.")
```

Sıra sizde #4 ÇÖZÜM

```
sayi1 = int(input("Bir sayi giriniz : "))

sayi2 = int(input("Bir sayi giriniz : "))

sayi3 = int(input("Bir sayi giriniz : "))

sayi4 = int(input("Bir sayi giriniz : "))

maxsayi = sayi1
```



DEVAMI

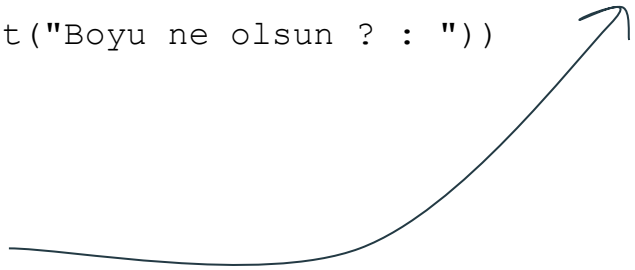
```
if (sayi2>maxsayi):
    maxsayi = sayi2
if (sayi3>maxsayi):
    maxsayi=sayi3
if (sayi4>maxsayi):
    maxsayi=sayi4
print("En büyük sayi : ",maxsayi)
```

Sıra sizde #5 ÇÖZÜM

```
a = int(input("İki basamaklı bir sayı giriniz :"))

b = a / 10

c = a % 10

b = int(b)

d = b * b + c * c

print("sonuç = " ,d)
```

Sıra sizde #6 ÇÖZÜM

```
en = int(input("Eni ne olsun ? : "))
boy = int(input("Boyu ne olsun ? : "))
a = 0
b = 0
```



DEVAMI

```
for a in range (boy) :
    if b % 3 == 0 :
    print("*"*en)
    elif b % 3 == 1:
    print("o"*en)
    else :
    print("-"*en)
    b = b + 1
```

ÇOK TEŞEKKÜR EDERİZ :)

Daha fazla çalışabilmeniz için kaynaklar;

→ Youtube kanalımız DSC Galatasaray'daki Python eğitim serimize göz atabilirsiniz.

→ Döngüler(Break ve Continue ile birlikte)

→ Sorularınız olması durumunda bizlere Instagram sayfamızdan veya web sitemizden ulaşabilirsiniz.

Hazırlayan: Bengü YURDAKUL