

# Python Programlama

## Ders 2

Ali Mertcan KOSE Msc.

`amertcankose@ticaret.edu.tr`

İstanbul Ticaret Üniversitesi



İSTANBUL TİCARET  
ÜNİVERSİTESİ

# Dizeler

Dizeler, büyüklüğe duyarlı karakterler dizisidir.

Harfler, özel karakterler, boşluklar, rakamlar olabilir.

Dizeler tırnak işaretleri veya tek tırnak işaretleri içine alır.

```
hi = "hellothere"
```

```
ch = 'a'
```

Dizeler sayısal değerler içerebilir, ancak değerler Dizelere olarak depolanır.

```
num = 57
```

```
numStr = "56"
```

num	int	1	57
numStr	str	1	56

Figure 1: Sayısal Dizeler.

Dizeler ==,<> gibi değerlerle karşılaştırılabilir

# Dize İşlemleri (String İşlemleri)-len()

Python, bir dizeye uygulanabilecek bir dizi işlev sunar.

Dizelerin uzunluğunu bulabilir, dizeler arayabilir, alt dizeleri bulabiliriz.

len(str), bir dizideki karakter sayısını döndürür.

```
In[3]:len("abc")
```

```
Out[3]:3
```

# Dize İşlemleri-Dizin Oluşturma

Bir dizedeki her karakterin bir dizini veya konumu vardır.

Python'da indeksleme, bir dizinin ilk ögesini belirtmek için sıfırdan başlar.

Köşeli parantezler, değeri belirli bir endekste konumda elde etmek için kullanılır.

```
s = "abc"
```

```
s[0] evaluates to 'a'
```

```
s[1] evaluates to 'b'
```

```
s[2] evaluates to 'c'
```

# Dize İşlemleri-Dizin Oluşturma

`s[3]` trying to index out of bounds, error

`s[-1]` evaluates to 'c'

`s[-2]` evaluates to 'b'

`s[-3]` evaluates to 'a'

index: 0 1 2 indexing always starts at 0

index: -3 -2 -1 last element always at index -1

# Dize İşlemleri-Dizin Oluşturma

Dizin oluşturma, bir dizeden tek tek karakterleri ayıklamak için kullanılabilir.

Verilen dizin dizinin sınırları dışındaysa, bir hata iletisi görüntülenir.

```
s[3]
```

IndexError:string index out of range

# Dize İşlemleri-Dizin Oluşturma

Negatif dizinler verilirse, indeksleme dizenin sonunda başlar.

```
s[-1]
```

```
Out[5]: 'c'
```

```
s[-2]
```

```
Out[6]: 'b'
```

```
s[-3]
```

```
Out[7]: 'a'
```

```
s[-4]
```

```
IndexError: string index out of range
```

Dilimleme, belirtilen uzunluktaki alt dizeleri ayıklamak için kullanılır.

Eğer  $s$  bir dize ise,  $s[\text{start} : \text{end}]$  ifadesi  $s$ 'nin dizin başlangıcında başlayan ve dizinde  $\text{start}$  ve bitiş indeksi  $\text{end} - 1$



# Dize İşlemleri-Dilimleme

Örnekler:

```
s = "hello world!"
```

```
s[0:len(s)]
```

```
Out[8]: 'hello world'
```

```
s[6]
```

```
Out[10]: 'w'
```

```
s[6:11]
```

```
Out[11]: 'world'
```

```
s[:]
```

```
Out[11]: 'hello world!'
```

# Dize İşlemleri-Dilimleme

Dize dilimlemesi kullanılarak [start : end : step]

iki numara vererseniz varsayılan olarak [start : end], step = 1 olur.

Ayrıca sayıları atlayabilir ve sadece kolon bırakılabilir.

```
s="abcdefgh"
```

```
s[3:6] -> evaluates to "def, same as s[3:6:1]
```

```
s[3:6:2]->evaluates to "df"
```

```
s[:]->evaluates to "abcdefgh", same as s[0:len(s):1]
```

```
s[::-1]-> evaluates to "hgfedcba", same as s[s-1:-(len(s)+1):-1]
```

```
s[4:1:-2]-> evaluates to "ec"
```

# Dize Birleştirme

Dize birleştirme operatörü (+): iki dizeyi birleştirir.

```
name = "ana"
```

```
name = 'ana'
```

```
greet = 'hi' + name
```

```
greeting = 'hi' + " " + name
```

Diğer türlerdeki verileri dizeyle birleştirmek için, önce str() işlevini kullanarak bir dizeye dönüştürmeniz gerekir.

```
"a" + 5
```

TypeError: must be str, not int

```
"a" + str(5) -> "a5"
```

# Dize Tekrarlama İşleci

Tekrarlama operatörü (\*) n bir tamsayı deperi ve s'nin bir dize olduğu n \* s ifadesi, s'nin n tekrarı olan bir string değişkeni olarak değerlendirilir.

3 \* 2'nin 2+2+2 ye eşit olması gibi, 3 \* "a" ifadesi de şuna eşdeğerdir.

"a + "a" + "a" ("aaa")

Python dökümanlarında tanımlandığı gibi bir dize üzerinde gerçekleştirilen işlemler örneği aşağıdaki gibidir.

```
silly = 'hi' + " " + name * 3
```

Error:

```
'a' * 'a'
```

TypeError: can't multiply sequence by non-int type 'str'

# Input

Python 3, doğrudan kullanıcıdan girdi almak için kullanılacak bir işleve sahiptir.

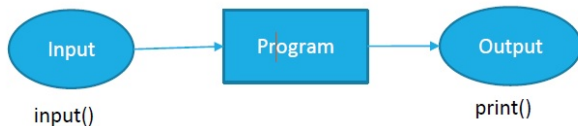


Figure 2: Input

Örnek:

```
text = input ("Type anything:") #Prints whatever is in quotes.
```

Type anything: hi #User types something, hits enter.

```
print (3*text) #Binds that value to a variable.
```

hihihi

# Giriş Dizeleri ve Tip Atama

input() fonksiyonu string değişkenine dönüşür

---

<sup>^</sup> Name	Type	Size	
age	str	1	30
name	str	1	Lisa

Figure 3: Input

# Giriş Dizeleri ve Tip Atama

Örnek:

```
name= input ("Enter your name:")
```

Enter your name: Lisa

```
print (type(name))
```

```
<class 'str'>
```

```
age = input ("Enter your age:")
```

Enter your age:30

```
print (type(age))
```

```
<class 'str'>
```



# Giriş Dizeleri ve Tip Atama

Sayılarla çalışıyorsanız, dönüş dizesini atamanız gerekir.

```
num=int(input("Type a number. . . ))
```

```
print (5*num)
```

# Input/Output Örneği:

Aşağıdaki yazdırma ifadelerinde birleştirme operatörünü kullanarak elde edilen dize ile virgülle elde edilen dize arasındaki farka dikkat ediniz.

```
name = input ("Enter your name:")
```

Enter your name: Lisa

```
print ("Are you really" + name + "?")
```

Are you really Lisa?

```
print("Are you really",name,"?")
```

Are you really Lisa ?

# Character Kodlaması

Bir bilgisayar için metin yoktur, sadece byte vardır.

Metin bir yorumdur, byte insan tarafından okunabilir biçimde görselleştirmenin bir yoludur. Ve byte yanlış yorumlanırsa, garip görünen karakter veya bir hata elde edersiniz.

Karakter kodlaması, byte'ı yorumlamanın belirli bir yoludur.

Örneğin, 97 değerine sahip bir byte'ın 'a' anlamına geldiğini söyleyen bir arama tablosudur.

Python karakter kodlama biçimini UTF-8 olarak varsayar.

Bir kaynak kodu kodlaması tanımlamak için, kaynak dosyalara dosyada birinci veya ikinci satır olarak sihirli bir yorum yerleştirilmelidir.

Örneğin;

```
# -*-coding:utf-8-*-or coding=<ISO-8859-9
```

Python build-in fonksiyonları: **ord()** ve **chr()**

- **ord(char)** karakterler için nümerik kodlama
- **chr(num)** integer sayılar için ilişkin karakterlerin kodlanması.

# Character Kodlaması

## Örnek

Bir çalışanın saatlik ücretini ve çalışılan saat sayısını giren ve çalışanın maaşını hesaplayan ve görüntöleyen bir komut dosyası yazın.

```
hourly_wage = float(input('Saatlik ücretinizi giriniz:'))
```

```
hours= int(input('günde kaç saat çalışıyorsunuz?'))
```

```
salary= hourly_wage * hours
```

```
print('Günlük maaliniz:' +str(salary+ 'TL'))
```

## Hızlı Çözüm:

Saatlik ücretiniz: 200

GÜnde kaç saat çalışıyorsunuz 8

Günlük maaşınız: 1600 TL

İfade yürütme sırasına akış kontrolü denir.

Aksi belirtilmedikçe, bir yöntem aracılığıyla ifade yürütme sırası birbiri ardına doğrusaldır.

Bazı programlama ifadeleri karar vvermemize ve tekrarlar yapmamıza izin verir.

Bu kararlar, doğru veya yanlış olarak değerlendirilen **boolean ifadelerine** (koşullar olarak da adlandırılır) dayanmaktadır.

# Koşullu ve Boolean İfadeleri

Koşullu bir ifade, daha sonra hangi ifadenin yürütüleceğini seçmenizi sağlar.

Koşullu ifadeler if/else ifadeleri kullanılarak uygulanır.

Boolean ifadesi, doğru ya da yanlış olarak değerlendirilen herhangi bir ifadedir.

Boolean ifadesi aritmetik ifadeleri, mantıksal ve ilişkisel operatörleri vb. uygulamaları içerebilir.

# Değerleri Karşılaştırma ve İlişkisel Operatörler

Bir koşul genellikle Python'un eşitlik operatörlerinden veya ilişkisel operatörlerinden birini kullanır ve bunların tümü boolean sonuçlarına döndürülür.

**== eşitlik**

**!= eşit değil**

**< az**

**> fazla**

**<= eşit veya az**

**>= eşit veya fazla**

## Not

Eşitlik operatörü (==) ve atama(=) operatörü arasındaki farka dikkat ediniz!



# Karşılaştırma Operatörleri (int,float,string)

i ve j'nin değişken adları olduğunu varsayalım.

Aşağıdaki karşılaştırmalar bir boolean olarak değerlendirilir.

Operator	Description
>	greater than
<	less than
==	equal to
<=	less or equal to
>=	greater or equal to
!=	lnot equal to

Figure 4: Karşılaştırma Operatörleri

# Karşılaştırma Operatörleri (int,float,string)

$i > j$

$i \geq j$

$i < j$

$i \leq j$

$i == j \Rightarrow$  eşitlik testi,  $i$  ve  $j$  aynı ise doğru

$i != j \Rightarrow$  eşitliksizlik testi,  $i$  ve  $j$  aynı değilse ise doğru

# Karşılaştırma Operatörleri- (in, not in)

Python'un dizelere de uygulanabilen iki operatörü vardır, içinde değil, bu operatörler bir gruba üyeliği test etmek için kullanılır.

Kullanımı:

Örnek

'way' in 'John Wayne'

Out[1]: False

'way'.lower() in 'John Wayne'.lower()

Out[2]: True

'was' not in 'T was happy'

'was' not in 'I'm happy'

Out[4]: True

# Mantık Operatörleri- (on bools)

A ve B'nin değişken adları (Boolean değerleri ile birlikte) olduğunu varsayalım.

not a -> Yanlış ise Doğru, Doğru ise Yanlış

a and b -> Her ikisi doğru ise Doğru, Her ikisi yanlış ise yanlış

a or b -> Bunlardan biri ya da her ikisi doğru ise Doğru, ikisinde doğru değilse yanlış

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Figure 5: Mantık Operatörü

# Python Operatör Önceliği

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR
= %= /= //= -= += *= **=	Assignment Operators

Figure 6: Operatör Önceliği

# Dallanma-if İfadeleri

if *< condition >*:

*< statement >*

*< statement >*

...

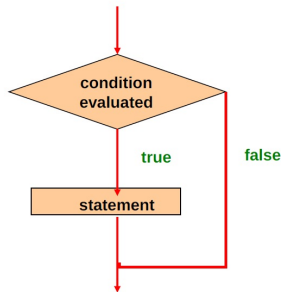
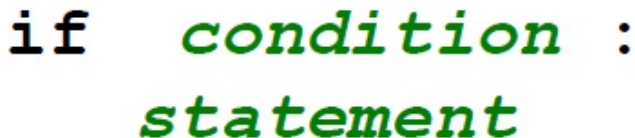


Figure 7: if ifadesi

if Pythona ayrılmış bir kelimedir.

koşul bir boolean ifadesi olmalıdır. DOğru veya yanlış olarak değerlendirilmelidir.

koşul doğru ise, ifade uygulanmalıdır. Yanlış ise, ifade geçilmelidir.



**if**    *condition* :  
*statement*

Figure 8: if Statement

Python'da Semantatic olarak anlamlılık.

Diğer programlama dillerinin çoğu boşlukları yok sayar ve girinti kodu okuyucu içindir.

Python, girintinin gerekli olması ve yürütme sırasında keskin bir şekilde uygulanması nedeniyle olağandışıdır.



Örnek:

```
x= 5
```

```
if x>10:
```

```
    print(x)
```

```
    print("done")
```

Output:

done

```
x= 5
```

```
if x>10:
```

```
    print(x)
```

```
    print("done")
```

Output: -

```
x= 5
```

```
if x>10:
```

```
    print(x)
```

Output: **ERROR**

# Dallanma-if İfadeleri

```
if (num%2) == 0:
```

```
    print ('num is even')
```

```
# non-zero values evaluate to True
```

```
if 5:
```

```
    print( "condition is true")
```

```
# zero values evaluate to false
```

```
if 0: print( "Condition is false" )
```

Bir çalışanın saatlik ücretini ve çalışan saat sayısını giren ve çalışanın maaşını hesaplayan ve görüntüleyen bir komut yazın. 40 saatten fazla çalışması durumunda çalışan 100\$ alacaktır.

```
hourly_wage: float (input('saatlik ücretinizi giriniz'))
```

```
hours=int(input('kaç saat çalıştınız'))
```

```
salary= hourly_wage*hours
```

```
if hours > 40:
```

```
    salary +=100
```

```
print('Maaşınız:'+str(salary)+'$')
```

Sonuç:

saatlik ücretinizi giriniz: 12.5

kaç saat çalıştınız: 50

Maaşınız: 725 \$

# if/else İfadeleri

```
if < condition>
```

```
<statement>
```

```
<statment>
```

```
...
```

```
else:
```

```
...
```

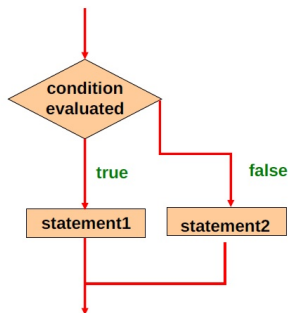


Figure 9: if/else komutu

# if/else İfadeleri

if-else ifadesi oluşturmak için if ifadesine else yan tümcesi eklenebilir.

if conditiona:

statement1

else:

statement2

Koşul doğruysa, ifade 1 yürütülür; Koşul yanlışsa, ifade2 yürütülür.  
Her iki ifade yürütülmez, biri veya diğer ifade yürütülür.

# Örnek

```
x,y = 2,3
```

```
if x>y:
```

```
    print('x is büyük)
```

```
else:
```

```
    print('x büyük değildir')
```

```
print ('Done')
```

Output:

x büyüktür

Done



Bir çalışanın saatlik ve çalışılan saat sayısını giren ve çalışanın maaşını hesaplayan ve görüntüleyen bir komut senaryosu yazınız. Çalışanın 40 saatin üzerinde çalıştığı her ekstra saat için %50 daha fazla alacaktır.

```
hourly_wage= float(input('saatlik ücretinizi giriniz'))  
hours=int(input('kaç saat çalışıyorsunuz'))  
if hours <40:  
    salary=hourly_wage * hours  
else:  
    salary=40*hourly_wage + (hours-40)*hourly_wage*1.5  
print ('maaşınız:'+str(maaşınız)+'$')  
saatlik ücretiniz:20  
kaç saat çalışıyorsunu 60  
maaşınız: 1400$
```

# İç içe if İfadeleri (if/elif)

if < *condition* >:

< *statment* >

< *statement* >

...

elif < *conditiona* >:

< *statement* >

< *statment* >

else:

< *statment* >

< *statement* >

...

# İç içe if İfadeleri (if/elif)

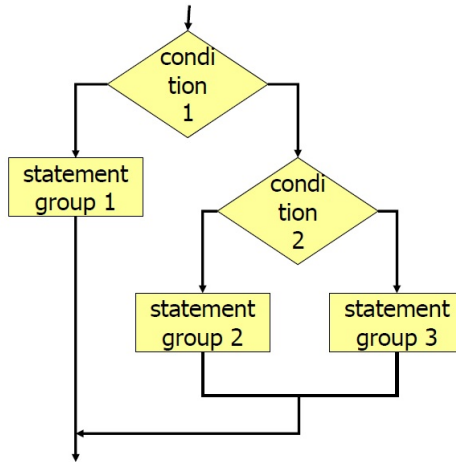


Figure 10: İç içe if/elif komutu

# Örnek

$x, y = 2, 2$

if  $x > y$ :

print ('x büyüktür')

elif  $x < y$ :

print('y büyüktür')

else:

print('x ile y eşittir.!!')

print('Done')

Output:

x ile y eşittir.

Done

## Öğrencilere Harf Notu verme Uygulaması

Range	Grade
$100 \geq \text{grade} > 90$	A
$90 \geq \text{grade} > 80$	B
$80 \geq \text{grade} > 70$	C
$70 \geq \text{grade} > 60$	D
$\text{grade} \leq 60$	F

# Alıştırmalar

- 1 Kullanıcıdan bir sayı girin ve sayı 5 veya 7 ye bölünürse çıkan sayısının 5 e ya da 7 e bölündüğünü yazarak elde edin.
- 2 1-9 arasında bir sayı başlatan ve kullanıcıyı numarayı tahmin etmesini isteyen bir Python programı yazın. Kullanıcı, yanlış tahmin ederse “Yanlış Tahmini” mesajını yazın, doğru tahmin ise “iyi tahmin” olarak iletin.
- 3 Bir kelime girin ve kelimenin başında ve sonunda aynı harfe sahip olup olmadığını uygun bir mesajla oluşturun.
- 4 Girdi olarak boy ve kilogram girerek Beden Kitle İndeksi Hesaplaması yapınız. Aşağıdaki değer aralıklarına göre sınıflama yapınız.

$bmi \geq 30 = \text{obez}$

$25 \leq bmi < 30 = \text{aşırı kilolu}$

$20 \leq bmi < 25 = \text{Normal}$

$bmi < 20 = \text{zayıf}$