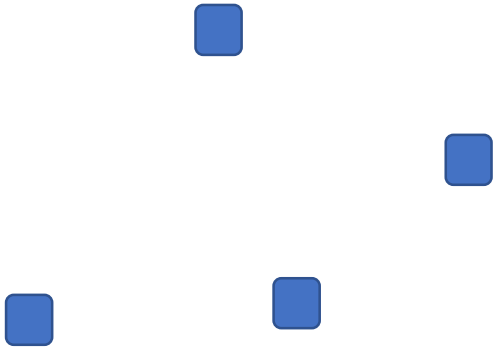
Değişken Tanımlama

Operatörler

Tip Dönüşümleri

GEÇEN HAFTADAN

- Neden Python
- Python Dili Özellikleri
- Python Dili Yapısı
- Python Interpreter Kurulumu
- Python IDE
- Python Paket Kurulumu
- Python I/O komutları



K O N U L A R

- Değişken kullanımı
- Operatörler
- Tip Dönüşümü
- İşlem önceliği
- Denklemlerin yazılışı

https://tr.sputniknews.com/karikatur/202003271041700421-sevdikleriniz-icin-evde-kalin/ (19/10/2020)

Python Değişken Tanımlama ve Atama Operatörü

Değişken tanımlama ve atama

Program içerisinde değeri değişebilen etiketlerdir. Pythonda değişkene tip tanımlamasına gerek yoktur dolayısıyla bir değişkene farklı türdeki (*int, double, complex, string* vb) değerler atanabilir.

değişken = değer

değişken = değer1 + değer2

değişken1, değişken2, değişken3 = değer1, değer2, değer3

değişken1= değişken2= değişken3= değer

List olarak

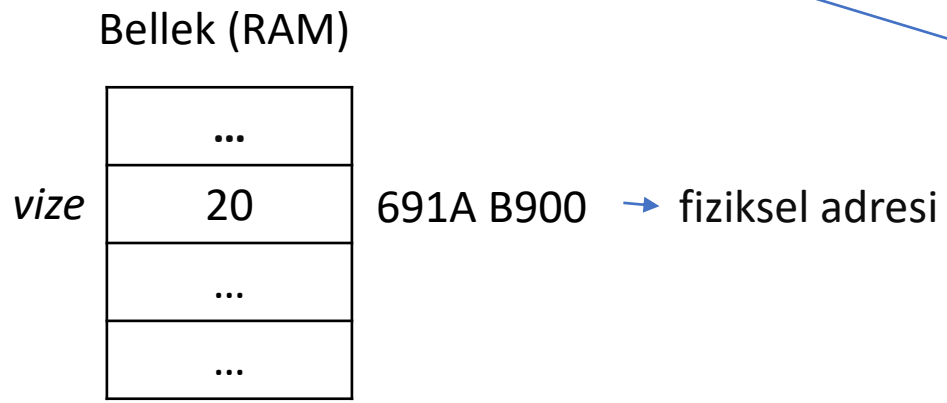
*değişken1, *değişken2 = değer1, değer2, değer3, değer4, değer5*

Python Değişken Tanımlama ve Atama Operatörü

Bütün değişkenler *class* yapısındadır ve bellekte nesne olarak tutulur

Örneğin : *vize = 85*

vize = 20




```
vize=85
print ( type(vize) )
print ( id(vize) )
```

```
<class 'int'>

93851093743872
```

Python Atama Operatörler

Değişken ismi verme kuralları

- İngiliz alfabesindeki 26 harf, (0-9) rakamlar ve "_" (alt tire) kullanılmalıdır.
- Dilin kendi komutlarını (reserved word) değişken ismi olarak kullanılamaz
- *İlk karakter rakam olamaz*
- *En az bir karakter olmalıdır*

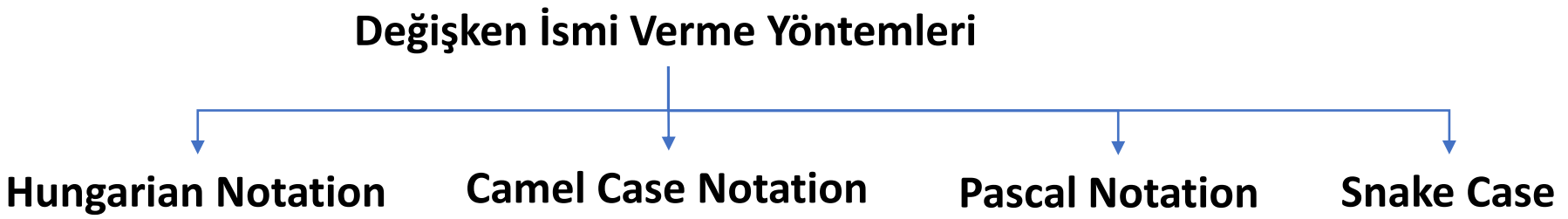
Python programı ekran çıktısı nasıl olur:

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

BİLGİ NOTU:

Python'da değişken tanımlarken küçük harflerin kullanılması tavsiye edilir

Genel Değişken Tanılama Yöntemleri



- ***Hungarian Notation (Macar Metodu)***

  Değişken ismi, değişkenin tipini ve içerisinde barındıracak bilgiyi çağrıştıracak şekilde verilmesidir.

  Örneğin: *int_vize= 75 veya vize_int=75*
- ***Camel Case Notation***

  Değişken isimi ilk karakter hariç diğeri kelimeler büyük harf ile başlar

  Örn : *vizeNotu=80,*
- ***Pascal Notation:***

  Değişken ismindeki tüm kelimelerin ilk karakteri büyük harf ile başlar

  Örn : *VizeNotu=90*
- ***Snake case Notation:***

  Değişken ismindeki kelimelerin arasına "_" alt tire konut. Küçük harflerle yazılır

  Örn : *vize_notu=90 ; oorenci_bilgi_durumu="başarılı";*

Değişken Tanımlama Yöntemleri

DOĞRU/YNALIŞ

adi_soyadi
Vize1
1vize
_final
devam
X
vize final
x+y

_
imCom@gamil.com
final-
anastasmumsatsana
class
@edu

Python Atama Operatörler

- ***Değişkenlerin alabileceği değerler***

- **None**
- **False/True**
- Numeric tipler Örn: 0, 0.0, 0j.
- Boş dizi : ( ), [ ]
- Boş String: '', ""
- Boş mapping: Örn: {}
- Kullanıcı tanımlı class'lar

```
x="None"
print(5*x)
```

```
x=None
print(5*x)
```

```
TypeError: unsupported operand type(s)
 for *: 'int' and 'NoneType'
```

Python Atama Operatörler

- ***type(...)*** komutu ile bir değişkenin tipini öğrenebiliriz
- ***isinstance(değer, tip)*** komutu ile değerin o tipte olup olmadığı araştırılır

```
print(type(3j))
<class 'complex'>

print (type(3.14))
<class 'float'>

print (type("3.14"))
<class 'str'>

print (type(4==5))
<class 'bool'>
```

```
print(type([3,6]))
<class 'list'>

print(type((3,5)))
<class 'tuple'>

print(type({3,5}))
<class 'set'>

print(type({"a":3,"b":5}))
<class 'dict'>

print(type(b'123'))
<class ‘bytes'>
```

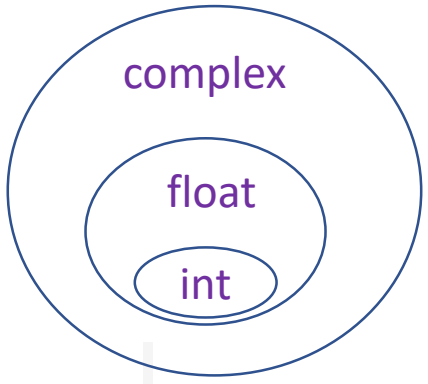
```
print(isinstance( 2,int))
print(isinstance( 2.0,float))
print(isinstance( 2.0j,complex))
print(isinstance( [2,4,5],list))
print(isinstance( {"a":2,"b":4},dict))
print(isinstance( {2,4,5},set))
print(isinstance( (2,4,5),tuple))
isinstance("slm",str)
isinstance(chr(65), str)
isinstance(ord('A'), int)
print(isinstance( b'dsdfg',bytes))
```

Python Tip Dönüşümü

Uygun olan tipleri birbirine dönüştürebiliriz bir

- int(…)
- float(…)
- complex(…)
- binary(…)
- str (…)

- list(…)
- set(…)
- tuple(…)
- dict(…)



ÖRNEK:

```
x=int(3.14)
y=float(2)
z=complex(3)
print(x,y,z)
```

```
3     2.0    (3+0j)
```

ÖRNEK:

```
X = complex("3.2j")
Y = X*5
print(Y)
# çıktı 16j
```

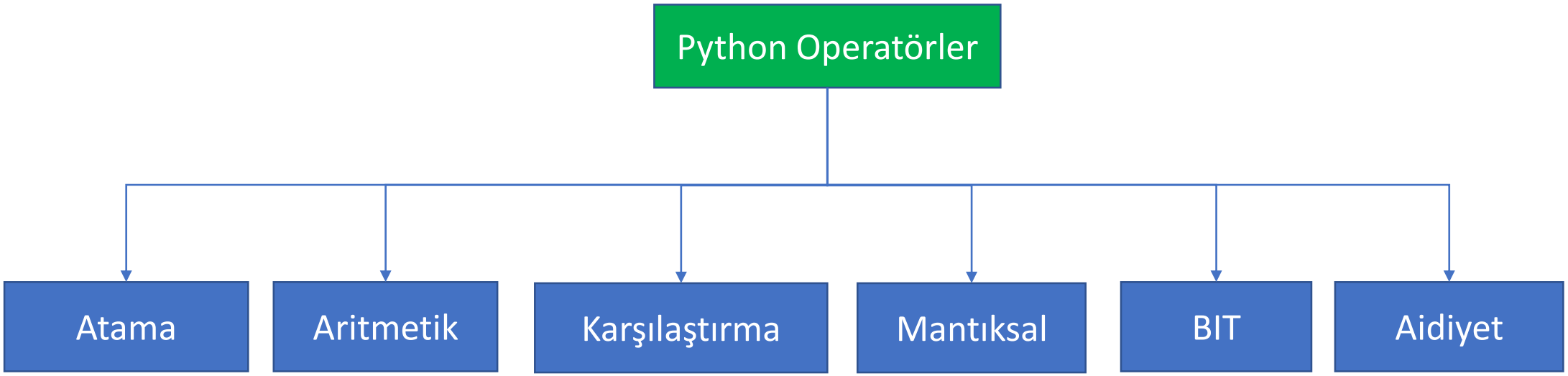
ÖRNEK:

```
vize=input("vize notu:")
vize_notu=(int(vize))*0.4
print(vize_notu)
```

ÖRNEK:

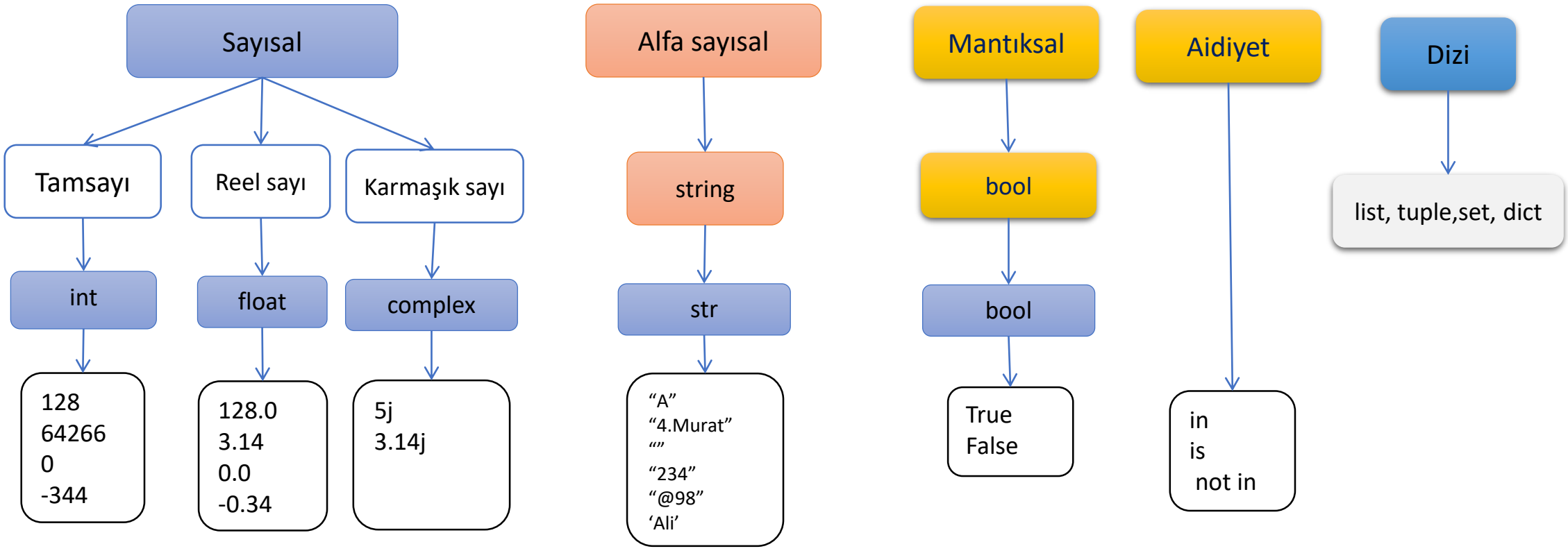
```
x=[1,2,3,4]
y=set(x)
z=tuple(x)
print(x,y,z)
# çıktı[1,2,3,4] {1,2,3,4} (1,2,3,4)
```

OPERATÖRLER



https://docs.python.org/3.1/library/stdtypes.html

Python Veri Tipleri



Atama Operatörleri

Sağdaki ifadeleri soldaki değişkenlere aktarır. Eğer sağ tarafta birden çok işlem varsa öncelikle bu işlemler yapılıp sonucu sol taraftaki değişkene aktarılır

x = 5; x = x+1

Operatör	Meaning (Anlamı)	Example (örnek)
=	Atama yap	X=5; z, y =6,8
+=	Topla ve ata	Y += 9
*=	Çarp ve ata	Y *=5
-=	Çıkar ve at	Y -=3
/=	Böl ve ata	Y /=2
%=	Mod al ve ata	Y %=5
//=	Tam böl ve ata	Y //=2
&=	Mantıksal AND yap ata	y &=5
>>=	Bir bit sağa kaydır ata	y >>=5
<<=	Bir bit sola kaydır ata	y <<=5
^=	XOR yap ata	y ^=5

Aritmetik Operatörleri

Operatör	Meaning (Anlamı)	Example (örnek)
+	Toplama, ekleme	X=5+8
-	Çıkarma	Y=15-9
*	Çarpma	Y=5*8
/	Bölme	Y=16/3 “’ 5.3333 “’
//	Tam Bölme	Y=16//3 # 5
%	Mod (kalanı verir)	Y=16%3 # 1
**	Üs alma	Y=3**2

```
print(5/3)      # 1.6666666666666667
print(4/2)      # 2.0
print(5//3)     # 1
print(5%3)      # 2
```

Karşılaştırma Operatörleri

Sonucu ***bool*** türünden (***True/False***) olan operatörler

Operatör	Meaning (Anlamı)	Example (örnek)
>	Büyük	if 5>6
>=	Büyük veya Eşit	if 5>=6
<	Küçük	if 5<6 :
<=	Küçük veya Eşit	if 5<=6
==	Eşit	if 2==3
!=	Farklı	if (3 != 3)

Mantıksal Operatörleri

İki karşılaştırma işlemini birlikte değerlendirmeye yarayan operatör. Sonucu ***bool*** türündendir (***True/False***)

Operatör	Meaning (Anlamı)	Example (örnek)
or	VEYA	`if(5 !=3 or 2==3): print("A")`
and	VE	`if(5 !=3 and 2==3): print("A")` `else: print("B")`
not	DEĞİL	`if( not(5==3)): print("A")`

Örn:

```
if(5==3 and 5==5 or 2==1 ):
   print("Evet")
else:
   print("Hayır")
```

Sonuç: `Hayır`

Mantıksal Operatörleri

İki karşılaştırma işlemini birlikte değerlendirmeye yarayan operatör. Sonucu ***bool*** türündendir (***True/False***)

Örn:

```
x=3;
if(x==3 or 5 ):
    print("Evet")
else:
    print("Hayır")
```

Sonuç: `Evet`

Örn:

```
x=3,4,5;
if(x==y==z):
    print("Evet")
else:
    print("Hayır")
```

Sonuç: `Hayır`

NOT: Python bu gibi karşılaştırmaları kabul eder.
Ancak programcılık açısından **uygun bulunmaz.**

BIT'sel Operatörleri

Binary olarak işlem yapmaya yarar (int için 32 bitliktir)

Operatör	Meaning (Anlamı)	Example (örnek) x=5 için
|	OR	y= x|2 print(y) #7
&	AND	y= x&2 print(y) #0
~ (Alt+126)	Complement (2 ye tümleyen: sayının negatifi)	y= ~x print(y) #-6
^	EXOR	y= x^2 print(y) #7
>>	Right Rotate	y= x>>2 print(y) #1 bölme
<<	Left Rotate	y= x<<2 print(y) #20 çarpma

```
x=5
print(bin(x)) #0b101
y=x|2
print(y) #7
```

```
x=5
print(bin(x)) #0b101
y=~x
print(y) #-6
```

Tersi

1 0 1 → 0 1 0

\+ 1 bir ilave

$-(1\ 1\ 0)_2 = -(6)_{10}$

BIT'sel Operatörleri

Dosya, sunucu, veritabanı vb. erişim izinleri belirlenir .

okuma, yazma, değiştirme, okuma-yazma-değiştirme, okuma-yazma vb.

Sisteme gelen kullanıcının hakkına göre işlem yapmasına izin verilir.
Kullanıcın hangi haklar sahip olduğunu ve hangi işlemleri yapmasına izin vermek için kullanıcı hakkı ile sistemdeki hak karşılaştırılıp buna göre işlem yapmasına izin verilir.
Bunun için maskeleme (mask) yapılarak ilgili bitlere bakılır

64	**32**	**16**	**8**	**4**	**2**	**1**	**2^n**
okuma	yazma	değiştirme	silme	Kullanıcı tanımlama	Yedek alma	geri yükleme	
1	1	1	1	1	1	1	128
1	0	0	0	0	0	0	64
1	0	0	0	0	1	0	66

```
izin=int(input("izin giriniz:"))
if(izin & 64) : print("Okuma")
if(izin & 66) : print("Okuma ve Yedek Alma")
if(izin & 128): print("Tüm izinler")
```

```
izin giriniz:66
```

```
Okuma
Okuma ve Yedek Alma
```

Aidiyet (üyelik) Operatörleri

Bir koleksiyon (liste, tuble, array, dictionary vb) içerisinde bir verinin olup olmadığını araştır. *bool* değer döndürür.

Operatör	Meaning (Anlamı)	Example (örnek)
in	Bu değer var mı?	`if "ari" in ("İngiltere de ari sokan kişi aranıyor")`
not in	bu değer yok mu?	`if "5" not in ("İngiltere de ari sokan kişi aranıyor")`

```
x=[5,6,2,7,8,9]
for k in x:
  print(k)
```

```
5
6
2
7
8
9
```

```
x = ["İzmir", "Ankara", "Çorum"]
print("ankara" in x)
```

```
False
```

PYTHON ESCAPE KARAKTERLERİ

Dil için anlam ifade eden karakterleri o anlamdan çıkarıp normal göründüğü karakter olarak kullanmayı sağlar.
Yada normal karaktere bir anlama katar.

Escape Sequence	Meaning
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)

```
print("\")
```

```
File "<ipython-input-60-dff1232de400>",
line 1 print("\")
SyntaxError: EOL while scanning string literal
```

```
print("\\")  # \
```

```
print("ali\t\tveli")  # ali		veli
```

```
print("ali\n\tveli")
```

```
ali
	veli
```

```
print ("deneme\newline")
```

```
deneme
ewline
```

```
print ("deneme\b")
```

```
denem
```

Kaynak: https://docs.python.org/3.10/reference/lexical_analysis.html#literals (11/09/2020)

Araştırma

* Karakterini Python’da hangi amaçlar için kullanırız?

https://treyhunner.com/2018/10/asterisks-in-python-what-they-are-and-how-to-use-them/#:~:text=The%20**%20operator%20allows%20us,arguments%20in%20a%20function%20call.&text=Functions%20in%20Python%20can't,an%20exception%20will%20be%20raised.

SORU VE GÖRÜŞLER