PYTHON MATEMATİKSEL İŞLEMLER

Python'da kütüphaneler olmadan da matematiksel işlemler yapabiliriz. İlerleyen günlerde daha ileri seviye işlemler yapacağız ve kütüphaneler (Pandas,NumPy) konusuna değineceğiz.

Python'da Toplama İşlemi (addition)

"+" operatörü ile toplama işlemi yapabiliriz.

```
print(4+2)
```

```
6
```

Python'da Çıkarma İşlemi (subtraction)

"-" operatörü ile çıkarma işlemi yapabiliriz.

```
print(5-2)
```

```
3
```

Python'da Çarpma İşlemi(multiplatication)

"*" operatörü ile çarçma işlemi yapabiliriz.

```
print(5 * 7)
```

```
35
```

Python'da Bölme İşlemi(division)

"/" operatörü ile bölme işlemi yapabiliriz.

```
print(8 / 7)
```

```
1.1428571428571428
```

Yukarıda yaptıklarımızı farklı bir şekilde değişken tanımlayarak da gerçekleştirebiliriz.

ÖRNEK:1

```
x=20
```

```
y=80
```

```
çarpma=(x*y)
```

```
print(çarpma)
```

```
1600
```

ÖRNEK:2

```
GELİR=87000
```

```
GİDER=5674
```

```
NET=(GELİR-GİDER)
```

```
print(NET)
```

```
81326
```

Python ÜSLÜ İFADELER(EXPONENTİAL)

- "**" operatörü ile üslü işlemi yapabiliriz.

```
print(8 ** 2)
```

```
64
```

- Pow() komutu

Üstel değerin hesaplanmasına izin veren bir komuttur. Fonksiyonun genel hali ;

```
pow(taban,üs)
```

Nasıl çalıştığına örnek üzerinde bakalım.

```
taban=8
üs=2
print("üstel değer:" ,pow(taban,üs))
```

```
üstel değer: 64
```

Bölme işlemini düşünürsek "Bölen, Bölünen, Kalan, Bölüm" kısımlarından oluşur. Python'da hangi operatörler ile bölüm ve kalana ulaştığımızı aşağıda görebilirsiniz.

% OPERATÖRÜ (modulus)

Bölme işlemindeki kalan bulma için kullanılan bir operatördür ve dört işleme dahil olarak ele alınır. % operatörü birinci sayıyı ikinci sayıya böler ve ikinci sayıdan kalanı yazar.

```
print(3%2)
```

```
1
```

// OPERATÖRÜ (Floor Division Operator)

Bölme işleminde birinci sayıyı ikinci sayıya böler ve kalan sayıyı yazar.

```
print(9//2)
```

```
4
```

ÖRNEK:3

```
X=67

Y=17

KALAN=(X%Y)

BÖLÜM=(X//Y)

print("kalan=",KALAN ,"bölüm=",BÖLÜM)
```

ÇIKTI:

```
kalan= 16 bölüm= 3
```

PYTHON VERİ TÜRLERİ (DATA TYPES)

Programı kullanırken veri tipleri önemlidir. Değişkene atadığımız veri farklı veri türlerinde depolanabilir. Farklı veri tiplerinde tanımlanan değişkenler farklı işler yapabilir. Genel olarak veri türlerini aşağıdaki tabloda görmekteyiz.

Metin Veri Türü (String)	**Str**
Sayısal Veri Türü (Numeric)	İnt,float,complex
Sıra Veri Türleri (Sequence)	list,tuple
Haritalama Veri Türü (Mapping)	Dict

SAYISAL VERİ TÜRLERİ

```
X=87 # int tanımlanır.
y=27.65 # float tanımlanır.
z=27j # complex tanımlanır
```

Pythonda hangi tip olduğunu anlamak için verilen komutun genel hali

```
print(type(istenilen veri tipi))
```

METİN VERİ TÜRÜ

```
print(type('büşra'))
```

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

SIRA VERİ TİPİ

Üç tip sıra veri tipini ele alırsak farklı gösterimler ile adlandırıldıklarını görmekteyiz

```
print(type([1, 2, 3]))
```

```
<class 'list'>
```

```
print(type({9.8, 3.14, 2.7}))
```

```
<class 'set'>
```

```
print(type((9.8, 3.14, 2.7)))
```

```
<class 'tuple'>
```

HARİTALAMA VERİ TÜRÜ

```
print(type({'name':'BÜŞRA'}))
```

```
<class 'dict'>
```

VERİ TÜRÜNÜ ALMA

typ() komutu ile verinin hangi tip olduğunu anlayabiliriz.

ÖRNEK:

```
A=57
B=47.555
c=5+9j
d="ASYA"
e=([1,2,3])
f=({9.2,5.1,6.4})
g=({'isim':'büşra'})
```

Bu şekilde verileri tanımlanır.

Çıktı:

```
print(type(a))
```

```
<class 'int'>
```

```
print(type(b))
```

```
<class 'float'>
```

```
print(type(c))
```

```
<class 'complex'>
```

```
print(type(d))
```

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

```
print(type(e))
```

```
<class 'list'>
```

```
print(type(f))
```

```
<class 'set'>
```

```
print(type(g))
```

```
<class 'dict'>
```