

5 Kasım 2017

Bilgisayar Bilimi

Kur1 Ders Notları

Temel Veri Tipleri ve Değişkenler

Ahmet Tuna POTUR

İçindekiler	i
1. Merhaba Dünya(Hello World)	1
1.1. Hello World Örnekleri	1
2. Yorum Satırları	1
3. Temel Veri Tipleri ve Değişkenler	2
3.1. Basit Veri Tipleri	2
3.1.1. Tam Sayı (int) Veri Tipi	2
3.1.2. Ondaklıklı Sayı (float) Veri Tipi	2
3.1.3. Basit Matematik İşlemleri	2
3.1.4. Karakter Verileri (string - str)	2
3.2. Değişkenler ve Değişken Tanımlama	2
3.2.1. print() Fonksiyonu	3
3.2.2. int(), float(), str() Fonksiyonları	3
3.2.3. type() Fonksiyonu	4
3.2.4. Değişkenlere İsim Verirken Dikkat Edilmesi Gereken Kurallar	4
3.2.5. Matematik İşleminin Sonucunu Değişkene Atama (Önce İşlem Sonra Atama)	5
3.2.6. Değişkenlerin Değerini Arttırma	5
3.2.7. Aynı Değere Sahip Değişkenler Tanımlama	5
3.2.8. Farklı Değere Sahip Değişkenler Tanımlama	6
3.2.9. İki Değişkenin Değerini Birbiriyle Değiştirme	6
3.3. Tip Dönüşümleri	7
3.3.1. Otomatik Tip Dönüşümleri	7
3.3.2. İsteğe Bağlı Tip Dönüşümleri	8
3.3.3. input() Fonksiyonu	10
3.4. Matematik Operatörleri	12
3.4.1. İşlem Önceliği	12
3.4.2. İşlem Gruplama Operatörü Parantez ()	13
3.4.3. Üs Alma (**)	13
3.4.4. Karekök, Küp Kök Alma	13
3.4.5. pow() Fonksiyonu İle Üs Alma	14
3.4.6. Tam Sayı Bölme (/)	14
3.4.7. Mod Alma Operatörü (%)	15
3.4.8. İşaret Değiştirme (-)	15

3.5. Örnekler ve Alıştırmalar	16
3.5.1. Örnekler	16
3.5.1.1. IDLE Yardımıyla Klavyeden Girilen İki Tam Sayının Toplamını	16
3.5.1.2. Pycharm Yardımıyla Klavyeden Girilen İki Tam Sayının Farkı	16
3.5.1.3. Kişilerin Yaşı	17
3.5.1.4. Çemberin Çevresi ve Alanı	17
3.5.1.5. Girilen İki Sayının Ortalaması	17
3.5.2. Alıştırmalar	18
3.5.2.1. IDLE Kullanılarak Yapılan Hesaplamalar	18
3.5.2.2. Öğrenci Bilgileri	18
3.5.2.3. Vücut Kitle Endeksi(BMI) Hesaplama	19
3.5.2.4. Araç Yakıt Tüketimi	19
3.5.2.5. İki Değişkenin Değerini Birbiriyle Değiştirme	19
3.5.2.6. Sonucu Tam Sayı Çıkan Bölme İşlemi	20
3.5.2.7. Önce İşlem Sonra Atama	20
3.5.2.8. İkinci Dereceden Bir Bilinmeyenli Denklemlerin Kökleri	20
3.5.2.9. Hipotenüs Uzunluğu Bulan Program	21
3.5.2.10. Girilen Saniye Değerini Saat Dakika Saniye Olarak Gösteren Program	21

1. Merhaba Dünya(Hello World)

Bilgisayarınızda programlama dilleri için gerekli yazılımları kurduysanız kurduğunuz yazılımları denemek için ekrana “Merhaba Dünya” veya “Hello World” yazdıran kodu çalıştırabilirsiniz. “Hello World” yazdıran kodu çalıştırmak üzerinde çalıştığınız programlama dilinin kod satırları hakkında size ön bilgi verebilir.

“<https://helloworldcollection.github.io/>” Sitesinde bir çok programlama dilli için “Hello World” kodu bulunmaktadır.

1.1. Hello World Örnekleri

Aşağıda çeşitli programlama dilleri için ekrana nasıl “Hello World” yazdırılacağı görülmekte.

C dilinde “Hello World”

```
#include

int main(void)
{
    puts("Hello, world!");
}
```

C# dilinde “Hello World”

```
class HelloWorld{
    static void Main(){
        System.Console.WriteLine (
            "Hello, World!");
    }
}
```

C++ dilinde “Hello World”

```
#include

int main()
{
    std::cout << "Hello, world!";
    return 0;
}
```

Java dilinde “Hello World”

```
class HelloWorld {
    static public void main( String
args[] ) {
        System.out.println(
            "Hello World!" );
    }
}
```

PHP dilinde “Hello World”

```
<?php echo '<p>Hello World</p>';?>
```

Swift dilinde “Hello World”

```
println("Hello, world!")
```

JavaScript dilinde “Hello World”

```
document.write('Hello, world!');
```

Python dilinde “Hello World”

```
print("Hello, world!")
```

2. Yorum Satırları

Program içerisinde kodlar üzerinde açıklama yapmak gerekiyorsa yorum satırları kullanılmalıdır. Yorum satırları Python tarafından görülmez ve çalıştırılmaz. Çok yorum satırı kullanmak yazdığınız kodun daha karmaşık olmasına yol açar. Fakat hiç yorum satırı kullanmadan kod yazarsanız kodunuzda ne yapmak istediğiniz zor anlaşılır.

```
>>> # Tekli Yorum Satırı
```

```
>>> """ Çoklu yorum satırı
içinde istediğiniz kadar
satır kullanabilirsiniz """
```

3. Temel Veri Tipleri ve Değişkenler

3.1. Basit Veri Tipleri

3.1.1. Tam Sayı (int) Veri Tipi

Pozitif ve negatif sayıları içeren kesirsiz sayıların tamamı tam sayılardır. **Integer** tam sayı tiplerinin İngilizcesi olduğunda Python’da Integer ifadesi kısaca **int** olarak kullanılır. Örnek: 32, -41, 0, 10 -190, 1923.

3.1.2. Ondalık Sayı (float) Veri Tipi

Ondalık sayılar iki tam sayının birbirine oranı ile ifade edilebilen sayılardır. Ondalık sayılar Python ‘da “float” olarak geçmektedir. Örnek: 3.14, -1.22, 7.05, 3.0, -5.96323, -4.0, 0.17,-0.23.

3.1.3. Basit Matematik İşlemleri

```
>>> # Toplama
>>> 3 + 5
8      → int

>>> #Çıkarma
>>> 7-12
-5      → int

>>> #Bölme
>>> 5/2
2.5     → float

>>> #Çıkarma
>>> 13-5
8      → int

>>> #Çarpma
>>> 17 * 3
51     → int

>>> #Bölme
>>> 6/2
3.0     → float
```

Yukarıda 4 işlem örneği verildi. Python ‘da bölüm her zaman **float** çıktığı için son işlemin sonucu **float** çıkar.

3.1.4. Karakter Verileri (string - str)

Tırnak(" ") içerisinde yazılan her ifade karakter verisi olarak algılanır. Python’da "", ' ' veya "" "" tırnak sembolleri kullanılabilir. "" "" birkaç paragraftan oluşan karakter verileri için kullanılır. Karakter verileri aritmetik işlemlere giremez. Python’da karakter tanımı için **string** kelimesinin kısaltması olan **str** tanımı kullanılır. Örnek: "Merhaba", "AHMET", "12322"

```
>>> #Tek tırnak ile
>>> 'Tuna POTUR'
'Tuna POTUR'

>>> #Çift Tırnak ile
>>> "Tuna POTUR"
'Tuna POTUR'

>>> #Üç Tırnak ile
>>> """Tuna POTUR"""
'Tuna POTUR'

>>> """üç tırnak
bir den çok satırlı metinler için kullanılır"""
```

3.2. Değişkenler ve Değişken Tanımlama

Bir değişken bir değer taşıyan isimdir. Değişkenler programın çalışması sırasında gerekli olan değerlerin değiştirilmesi ve güncellenmesi amacıyla bellekte depolanması için kullanılır. Bir değişkene bir değer atamak için atama ifadesi **eşittir** “=” operatörü kullanılır. Diğer programlama dillerinin aksine Python’da değişkenleri önceden tanımlamanız gerekmez. Python’da değişkene değer atadığınızda değişken değeri ile birlikte oluşturulur.

```
>>> x
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in
<module>
  x
NameError: name 'x' is not defined

>>> x = 0
>>> x
0
```

Python’da değişken değer almadan oluşturulamaz. **x** değişkenine değer vermeden oluşturmaya çalıştığımızda hata oluşacaktır. **x** değişkenini oluşturmak için **x = 0** gibi bir tanımlama yapılması zorunludur.

```
>>> # Değişken ismi ve Değişkenin değeri
>>> yaş = 16
>>> boy = 1.85
>>> adı = "Can"
>>> print("Adınız:",adı,"\n","Yaşınız:",yaş,"\n","Boyunuz:",boy,"\n")
Adınız: Can
Yaşınız: 16
Boyunuz: 1.85
```

Değişkenler taşıdıkları verinin tipindedir.

Örnekte görüldüğü gibi değişkene hangi tipte değer verildiyse değişken o tipte oluşturulur.

- **yaş** değişkenine **16** tam sayısı girildiği için **yaş** değişkeni **int**,
- **boy** değişkenine **1.85** kesirli sayısı girildiği için **boy** değişkeni **float**,
- **adı** değişkenine **"Can"** karakter değeri atandığı için **adı** değişkeni **str** tipinde oluşturuldu.

3.2.1. print() Fonksiyonu

Fonksiyonlar konusunu ilerde detaylı bir şekilde göreceğiz ama bazı fonksiyonları şimdiden tanımamız gerekiyor. **print()** fonksiyonu parantez içine girilen değeri ekrana yazdırılır. Üstteki örnekte

print("Adınız:",adı,"\n","Yaşınız:",yaş,"\n","Boyunuz:",boy,"\n") ifadesinde

- **"\n"** satır karakteri bir satır alta geçmek,
- **"Adınız:", "Yaşınız:"** ve **"Boyunuz:"** karakter verileri doğrudan ekrana yazdırılmak,
- **adı, yaş** ve **boy** değişkenleri içinde tutulan değerlerin ekrana yazdırılması, için fonksiyona girildi.

3.2.2. int(), float(), str() Fonksiyonları

int() tam sayı, **float()** ondalıklı sayı, **str()** karakter fonksiyonları kendi tiplerinde değişkenleri oluşturmak ve değer olarak aldıkları verileri kendi tiplerinde verilere dönüştürmek için kullanılır.

```
>>> tam_sayı = int()
>>> ondalıklı = float()
>>> karakter = str()

>>> tam_sayı
0
>>> ondalıklı
0.0
>>> karakter
''
```

Bazen değişkenleri varsayılan değerleri ile tanımlamak isteyebilirsiniz. Üstteki gibi yapılan tanımlamada

- **tam_sayı** değişkeni **int** tipinde **0**
- **ondalıklı** değişkeni **float** tipinde **0.0**
- **karakter** değişkeni **str** tipinde **' '** değeri ile oluşturuldu

```
>>> # float => int çevirme
>>> int(3.14)
3
>>> int(-5.27)
-5
>>> pi = 3.14
>>> int(pi)
3

>>> # str => int çevirme
>>> int("65")
65
>>> a = "97"
>>> int(a)
97

>>> # int => float çevirme
>>> float(7)
7.0
>>> float(-9)
-9.0
>>> a = 53
>>> float(a)
53.0

>>> # str => float çevirme
>>> float("37.43")
37.43
>>> b = "23.87"
>>> float(b)
23.87
```

int(), float(), str() fonksiyonları değer olarak aldıkları verileri kendi tiplerine dönüştürür. Bu konu İsteğe Bağlı Tip Dönüşümleri konusunda detaylı bir şekilde işlenecek.

3.2.3. type() Fonksiyonu

```
>>> type(16)
<class 'int'>

>>> type(1.85)
<class 'float'>

>>> type("Can")
<class 'str'>
```

type(16) işlemiyle 16 değerinin **int**
type(1.85) işlemiyle 1.85 değerinin **float**
type("Can") işlemiyle "Can" değerinin **str**
olduğu program içerisinde öğrenilebilir.

```
>>> yaş = 16
>>> type(yaş)
<class 'int'>

>>> boy = 1.85
>>> type(boy)
<class 'float'>

>>> adı = "Can"
>>> type(adı)
<class 'str'>
```

type(yaş) işlemiyle yaş değişkeninin **int**
type(boy) işlemiyle boy değişkeninin **float**
type(adı) işlemiyle adı değişkeninin **str**
olduğu program içerisinde öğrenilebilir.

Değişkenler taşıdıkları verinin tipindedir. type(değişken) şeklinde çalıştırılan type() fonksiyonu bize değişkenin hangi tipte olduğunu döner.

```
>>> yaş = 16
>>> type(yaş)
<class 'int'>
```

```
>>> yaş="on altı"
>>> type(yaş)
<class 'str'>
```

```
>>> yaş = 15.5
>>> type(yaş)
<class 'float'>
```

```
>>> yaş = 15
>>> type(yaş)
<class 'int'>
```

Bir değişkenin hem değeri hem de tipi kod akışı içerisinde değiştirilebilir.

- yaş değişkenine 16 değeri girilerek yaş değişkeni **int** tipinde tanımlandı.
- Sonradan yaş değişkenine "on altı" değeri girilerek yaş değişkeni **str** tipine dönüştürüldü.
- Daha sonra yaş değişkenine 15.5 değeri atanarak yaş değişkeni **float** tipine dönüştürüldü.
- Son olarak yaş değişkenine 15 değeri atanarak yaş değişkeni **int** tipine dönüştürüldü.

Değişkenleri oluşturduğumuz tiplerle kullanmalıyız. Değişkenin program çalıştığı sürece belli bir anlamı ve rolü olduğu için tipi değişmemelidir. Çok nadir durumlar dışında değişkenlerin tipini değiştirmekten kaçınmalıyız.

3.2.4. Değişkenlere İsim Verirken Dikkat Edilmesi Gereken Kurallar

1. Değişkene içerdiği değer ile tutarlı isimler veriniz.
2. Değişkenlere isim verirken boşluk kullanmayınız.
3. Değişken isimleri sayı ile başlayamaz. Değişken isimleri bir karakter ile başlamalıdır.
4. Python3 de değişken adı olarak kullanabileceğimiz karakterler:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzÇĞİÖŞÜçğışöşü_0123456789
5. Programlama dillerinde kullanılan komut isimleri değişken olarak kullanılamaz.

Python' da Değişken Olarak Kullanılamayan Komut İsimleri								
and	as	assert	break	class	continue	def	del	elif
else	except	False	finally	for	from	global	if	import
in	is	lambda	None	nonlocal	not	or	pass	raise
return	True	try	while	with	yield			

6. Değişken isimlendirmelerinde boşluk karakteri yerine alt çizgi (_) karakteri kullanılabilir.
Örnek: tc_Kimlik_No
7. Değişken isimleri kelimeler arasına boşluk koymadan ilk kelimedenden sonra gelen kelimelerin baş harfleri büyük yazılarak oluşturulabilir. Buna "Camel Karakter" kullanımı denir.
Örnek: tcKimlikNo

8. `+*/%:;'"<>?|\()!@#$$%^&~` gibi özel karakterler değişken ismi içinde kullanılamaz.

(Sadece “_” sembolü kullanılabilir).

9. Değişken isimleri büyük küçük harf duyarlılığına sahiptir.

```
>>> TELEFON = "417 00 01"
>>> telefon = "415 11 12"
>>>
>>> TELEFON
'417 00 01'
>>> telefon
'415 11 12'
```

TELEFON = “417 00 01” ve telefon = “415 11 12” iki ayrı değişkenlerdir.

Hatalı Değişken İsimleri Örnekleri; 1sayı, ali@veli, +yaş, cc.com, Okul No, soru?, ara-toplam, and, print ...

3.2.5. Matematik İşleminin Sonucunu Değişkene Atama (Önce İşlem Sonra Atama)

```
>>> #Önce işlem sonra atama
>>> x = 3 + 5*2
>>> print("X değeri :",x)
X değeri : 13
```

$$x = 3 + 5*2$$

13

x değişkenine değer atamadan önce $3 + 5*2$ işlemi yapılmış. Python ‘da eğer değişkene değer atanmadan önce işlem yapıldıysa, önce eşittir operatörünün sağındaki işlem yapılır sonra sonuç değişkene atanır.

```
# Değişken ile Önce işlem sonra atama
>>> kütle = 100
>>> hacim = 25
>>> özkütle = kütle / hacim
>>> print("Özkütle =",özkütle)
Özkütle = 4.0
```

$$\begin{aligned} \text{kütle} &= 100 \quad \text{hacim} = 25 \\ \text{özkütle} &= \text{kütle} / \text{hacim} \\ &4.0 \end{aligned}$$

Bu örnekte **özkütle** değişkenine **kütle/hacim** işleminin sonucu atanmıştır. Sayısal verilerde olduğu gibi değişkenlerde de önce işlem yapılır sonra sonuç değişkene atanır. Kullanıcıdan değeri alıp kod içerisinde tanımladığımız denklem sonucunu değişkene atayan kodları çok kullanacağız

3.2.6. Değişkenlerin Değerini Arttırma

```
>>> #Önce işlem sonra atama
>>> x = 5
>>> x = x*2 + 3
>>> print("X değeri :",x)
X değeri : 13
```

$$\begin{aligned} x &= 5 \\ x &= x*2 + 3 \\ &13 \end{aligned}$$

Burada “bir matematik kuralını mı ihlal ettik?” sorusu akla gelebilir. Programlama içerisinde bu tip işlemler doğrudur ve fazlasıyla yapılır. Eşittir operatörünün kullanımında önce sağ taraftaki işlem yapılır. x değişkeni eşitliğin sağında değer gibi işleme girer ve sonuç x değişkenine atanır.

3.2.7. Aynı Değere Sahip Değişkenler Tanımlama

```
>>> # kod içerisinde aynı değere sahip değişkenler tanımlama
>>> a = 5
>>> b = 5
>>> c = 5
>>> print("a =",a,"b =",b,"c =",c)
a = 5 b = 5 c = 5
```

Kod içerisinde aynı değere sahip değişkenler tanımlamaya ihtiyaç duyabilirsiniz. Değişkenlere tek tek aynı değeri vererek tanımlayabilirsiniz.

```
>>> # kolay yoldan aynı değere sahip değişkenler tanımlama
>>> a = b = c = 9
>>> print("a =",a,"b =", "c =",c)
a = 9 b = c = 9
```

Python' da `a = b = c = 9` şeklinde tanımlaya yaparak aynı değere sahip değişkenleri bir seferde tanımlayabilirsiniz.

```
>>> ocak = mart = mayıs = temmuz = ağustos = ekim = aralık = 31
>>> nisan = haziran = eylül = kasım = 30
>>> şubat = 28
```

Yıl içerisinde aynı değere sahip birçok ay bulunmakta. Hangi ayın kaç gün çektiğini değişkelere tanımlamak için aynı değere sahip değişkenleri üstte görüldüğü gibi kolay bir şekilde tanımlayabilirsiniz.

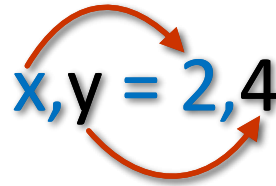
3.2.8. Farklı Değere Sahip Değişkenler Tanımlama

```
>>> a = 3
>>> b = 5
>>> print("a=",a," b=",b)
a= 3 , b= 5
```

Farklı değere sahip birçok değişkene tek tek değer vererek atama yapabilirsiniz.

```
>>> x,y=2,4
>>> print("x =",x," y =",y)
x = 2 , y = 4
```

`x,y = 2,4` atama işlemiyle bir anda bir çok değişkene farklı değer atayabilirsiniz.



3.2.9. İki Değişkenin Değerini Birbiriyle Değiştirme

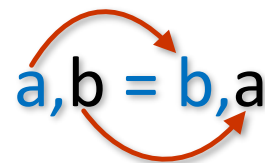
```
>>> a = 3
>>> b = 5
>>> a = b
>>> print("a =",a," b =",b)
a = 5 , b = 5
```

`a` ve `b` değişkenlerinin değerlerini bir biriyle değiştirmek gerekebilir. `a = b` gibi bir eşitlik yaparsak `b` değişkeninde bulunan değer `a` değişkenine geçecektir ve `a` değişkeni `b` değişkeni ile aynı değere sahip olacaktır. Bu durum istediğimiz sonucu vermez.

```
>>> a = 3
>>> b = 5
>>> takas = a
>>> a = b
>>> b = takas
>>> print("a =",a," b =",b)
a = 5 , b = 3
```

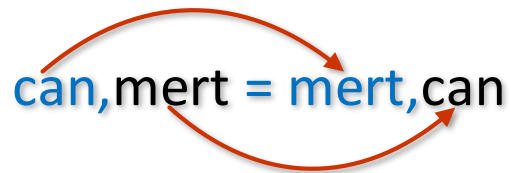
İki değişkenin değerini birbiriyle değiştirmek için `takas` değişkenine `a` değişkeni atanır, `a` değişkenine de `b` değişkeni atanır. Son olarak `b` değişkenine `takas` değişkeni atanır.

```
>>> a = 3
>>> b = 5
>>> a,b = b,a
>>> print("a =",a," b =",b)
a = 5 , b = 3
```



Python `takas` işlemi yapmak için bize kolay bir yol sunar. `a,b = b,a` yapmak iki değişkenin değerlerinin bir biriyle değiştirilmesini sağlar.

```
>>> can = "Müdür"
>>> mert = "Öğretmen"
>>> can,mert = mert,can
>>> print("Can =",can," Mert =",mert)
Can = Öğretmen , Mert = Müdür
```



İki değişkenin değerini nerede değiştireceğimize örnek olarak bir okulda müdür ve öğretmen olan iki kişinin görevlerinin değişmesi verilebilir. Can Müdür olarak görev yaptığı okulunda müdürlükten ayrılıp öğretmenliğe geçebilir. Mert' de öğretmenlikten müdürlüğe geçebilir. Can' ın müdürlük görevi Mert'e, Mert' in öğretmenlik görevi Can' a verecektir. Bu görev değişikliğini en iyi şekilde değişkenlerin değerleri değiştirilerek yapılır.

3.3. Tip Dönüşümleri

3.3.1. Otomatik Tip Dönüşümleri

```
#iki int sayı toplamı
>>> 3 + 5
8
>>> type(3 + 5)
<class 'int'>
```

İki **int** sayı arasında yapılan işlemin sonucu yine **int** tipinde olacaktır.

```
#float iki sayı toplamı
>>> 2.25 + 3.75
6.0
>>> type(2.25 + 3.75)
<class 'float'>
```

İki **float** sayı arasında yapılan işlemin sonucu tam sayı çıksa bile sonucun tipi her zaman **float** olur.

```
#int ve float iki sayı toplamı
>>> type(2)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> 2 + 3.14
5.140000000000001
>>> type(2 + 3.14)
<class 'float'>
```

float ve **int** sayı arasında işlem yapılırsa sonuç her zaman **float** sayı tipinde olur.

```
#int ve float iki değişkenin toplamı
>>> a = 12
>>> type(a)
<class 'int'>
>>> b = 23.5
>>> type(b)
<class 'float'>
>>> a + b
35.5
>>> type(a+b)
<class 'float'>
```

int ve **float** değişkenlerin toplamı yine **float** tipinde olacaktır.

```
#float iki değişken toplamı
>>> a = 12.3
>>> b = 23.7
>>> a + b
36.0
>>> type(a + b)
<class 'float'>
```

İki **float** değişken arasında yapılan işlemin sonucu tam sayı çıksa bile sonucun tipi her zaman **float** olur.

Görüldüğü gibi **float** ve **int** sayılar toplandığında sonuç her zaman **float** çıkar. Aslında programlama dillerinde iki farklı sayı tipi işleme giremez. Bunun sebebi **int** ve **float** gibi sayıların işlemci üzerinde farklı şekilde işlenmesi ve hafızada farklı miktarda alan kaplaması gibi donanımsal sebeplerdir. Fakat birçok programlama dilinde olduğu gibi Python’ da **int** ve **float** sayılarla yapılan işlemlerde **int** sayıların dönüşümlerini otomatik olarak yapar. Python **int** sayı tipindeki veriyi arka planda **float** sayı tipine yükseltir ve işlemi **float** sayılar arasında yapar. Bu yüzden **float** sayılarla yapılan işlemlerin sonuçları her zaman **float** çıkar.

Özetle; **float** ve **int** sayılarla işlem yapıldığında **int** sayılar **float** sayı tipine yükseltilir ve işlem **float** sayılar arasında yapıldığından işlemin sonucu her zaman **float** çıkar.

```
#sonucu tam sayı çıkan bölme işlemi
>>> 12 / 4
3.0
>>> type(12 / 4)
<class 'float'>
```

Python tip güvenliğini sağlamak için bölme işleminin sonucu **int** çıksa bile sonucu her zaman **float** çıkar. Bunun sebebi bölme işlemlerinin sonuçlarının çoğunlukla **float** çıkmasından kaynaklanmaktadır.

3.3.2. İsteğe Bağlı Tip Dönüşümleri

int(), **float()**, **str()** fonksiyonlarının değer olarak aldıkları verileri kendi tiplerine dönüştürdüğünü **int()**, **float()**, **str()** Fonksiyonları konusunda önceden görmüştük.

<pre>>>> # ondalıklı sayıyı >>> # tam sayıya çevirme >>> int(3.14) 3 >>> int(-2.79) -2 >>> pi = 3.14 >>> int(pi) 3</pre>	<pre>>>> # tam sayıyı >>> # ondalıklı sayıya çevirme >>> float(7) 7.0 >>> float(-9) -9.0 >>> a = 53 >>> float(a) 53.0</pre>
--	---

float sayıyı tamsayıya çevirmek için **int()** fonksiyonu kullanılır. **float** ‘dan **int** sayıya çevrim yapıldığında ondalıklı sayının tam kısmı değer olarak kullanılır.

Tam sayıyı ondalıklı sayıya çevirmek için **float()** fonksiyonu kullanılır. Tam sayıların kesir hanesi olmadığı için tam sayılar ondalıklı sayıya dönüştürüldüğünde **5.0** gibi bir değer oluşur.

<pre>>>> # tam ve ondalıklı sayıyı string’ e çevirme >>> str(47) '47' >>> a = 63 >>> str(a) '63'</pre>	<pre>>>> str(3.141592) '3.141592' >>> pi = 3.141592 >>> str(pi) '3.141592'</pre>
--	---

Sayıları karaktere çevirmek için **str()** fonksiyonu kullanılır. **str()** fonksiyonuyla sayıyı oluşturan tüm rakamlar veya nokta birer karaktere dönüştürülür.

```
# string veriyi tam ve ondalıklı sayıya çevirme
>>> int("93")          >>> float("3.141592")
93                     3.141592
>>> int("-17")         >>> float("-22.17")
-17                    -22.17
```

Bir string'i, tamsayıya çevirmek için `int()`, ondalıklı sayıya çevirmek için `float()` fonksiyonu kullanılır.

```
>>> # sayısal ifadenin yanında karakter kullanılıyor
>>> int("ahmet1981")
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    int("ahmet1981")
ValueError: invalid literal for int() with base 10: 'ahmet1981'
>>> # tip dönüşümü hatasının düzeltilmesi
>>> int("1981")
1981
>>> # karakter ifadesinin içinde iki kere nokta karakteri kullanılıyor
>>> float("3.14.324324")
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    float("3.14.324324")
ValueError: could not convert string to float: '3.14.324324'
>>> # tip dönüşümü hatasının düzeltilmesi
>>> float("3.14324324")
3.14324324
```

Karakterler sayısal veriye çevrileceği zaman tip dönüşümü yapılacak verinin sayısal karşılığı olması gerekir. string 'den int' e çevrilirken oluşan hatanın nedeni sayısal ifade ile sayısal olmayan karakter kullanılmasıdır. string 'den float' a çevrilirken oluşan hatanın sebebi karakter ifadesinin içinde iki kere nokta kullanılmasıdır.

```
>>> # string verilerin sayısal veriye çevrilerek işleme sokulması
>>> 71 + int("17")      >>> 12 + float("2.17")
88                     14.17
>>> int("28") + 37      >>> 19.23 + float("-5.5")
65                     13.73
```

Bu örnekte `int` ve `float` sayı değerlerinin `string` tipli değerler ile matematiksel işlemleri görülüyor. `string` veriler tip dönüşümüyle sayısal verilere dönüştürülerek `int` ve `float` sayılarla işlem yapılabilir.

```
>>> # string ve int değişkenlerin işleme girmesi
>>> a = 9
>>> b = "11"
>>> c = 3
>>> a + b + c
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    a + b + c
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> a + int(b) + c # int(b) ile string değer int'e geçirilerek işlemi giriyor
23
```

Bu örnekte `string` `b` değişkeni ve `int`. `a` ve `c` değişkenleri ile `a + b + c` işlemi yapılmış. `int` ve `string` değişkenler birlikte işleme giremeyecekleri için tip hatası oluşur. Bu hatanın çözümü için `int(b)` işlemiyle `b` değişkenin değeri `int`'e dönüştürülür.

```
>>> # string değişkenin değerini int'e çevirip başka değişkene atama
>>> x = "33"
>>> y = int(x) # x'in string değeri int'e çevrilerek y'ye atanıyor
>>> t = y * 2
>>> t
66
>>> print("x,y,t sırasıyla",type(x),type(y),type(t),"tipindedir")
x,y,t sırasıyla <class 'str'> <class 'int'> <class 'int'> tipindedir
```

String değere sahip **x** değişkeninin değeri **int**'e dönüştürülerek **y**'ye atandığı için **y** matematiksel işleme girebilir.

3.3.3. input() Fonksiyonu

Fonksiyonu kullanıcıdan değer almak için kullanılır. **input()** fonksiyonuna girilen değer kullanıcıya mesaj verir. **input()** fonksiyonu kullanıcının girdiği bilgiyi **string** olarak döner.

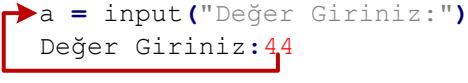
```
>>> input()
33
'33'
>>> input("Bir Değer Giriniz:")
Bir Değer Giriniz:47
'47'
```

input() fonksiyonu işletildiği anda ekranda beliren imleç bilgi girilmesini bekler.

```
>>> a = input("Değer Giriniz:")
Değer Giriniz:44
>>> a
'85'
>>> type(a)
<class 'str'>
```

Kullanıcıdan aldığınız değeri bir değişkene atamazsanız program içerisinde kullanıcıdan gelen değer kullanılamaz. **input()** fonksiyonuna girilen değer kullanıcıya değer girmesi için yönlendirici nitelikte mesaj verecektir.

```
>>> a = input("Değer Giriniz:")
İşlemlerle kullanıcıdan alınan string değer a değişkenine atanır.
```



```
>>> a = input("Değer Giriniz:")
Değer Giriniz:44
>>> print("Kullanıcının Girdiği Değer:", a, "Değerin Tipi:",type(a))
Kullanıcının Girdiği Değer: 44 Değerin Tipi: <class 'str'>
>>> b = input("Değer Giriniz:")
Değer Giriniz:ahmet1981
>>> print("Kullanıcının Girdiği Değer:", b, "Değerin Tipi:",type(a))
Kullanıcının Girdiği Değer: ahmet1981 Değerin Tipi: <class 'str'>
```

Görüldüğü gibi **input()** fonksiyonu kullanıcıdan **string** tipinde veri alır.

```
>>> a = input("Değer Giriniz:")
Değer Giriniz:8
>>> a + 2
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    a + 2
TypeError: Can't convert 'int' object to str implicitly
>>> type(a)
<class 'str'>
```

input() fonksiyonu kullanıcıdan **string** tipte değer aldığı için kullanıcıdan aldığınız değeri doğrudan bir değişkene atarsanız değişkenin tipi **string** olacaktır. Bildiğiniz gibi **string** tipte değişken ile **float** ve **integer** değişken işleme giremez.

```
>>> a = int(input("Değer Giriniz:"))
Değer Giriniz:8
>>> a + 2
10
```

Kullanıcıdan aldığınız değerin **int** ve **float** sayılarla işleme girmesi için **input** fonksiyonunu **int()** veya **float()** fonksiyonun içinde kullanmanız gerekir. Örnekte kullanıcıdan aldığınız **string** değer **int()** fonksiyonuyla tam sayıya dönüştürülür. Tam sayıya dönüştürülen kullanıcı verisi işleme girebilir.

```
>>> r = float(input("Çemberin Yarıçapını Girin:"))
Çemberin Yarıçapını Girin:5.25
>>> alan = 3.14 * r * r
>>> print("Çemberin Alanı:",alan)
Çemberin Alanı: 86.54625
```

Bu örnekte kullanıcı çemberin yarıçapını **float** değer olarak giriyor. **float** sayılar arasında işlem yapmak için **input()** fonksiyonu **float()** fonksiyonunun içinde kullanılmış. Böylelikle kullanıcının **input()** ile girdiği **string** değer **float()** fonksiyonuyla **float** değere dönüştürülüyor.

```
>>> yaş = int(input("Öğrencinin Yaşını Giriniz:"))
Öğrencinin Yaşını Giriniz:16
>>> boy = float(input("Öğrencinin Boyunu Giriniz:"))
Öğrencinin Boyunu Giriniz:1.85
>>> ad = input("Öğrencinin Adını Giriniz:")
Öğrencinin Adını Giriniz:Can
>>> print("Adı:",ad,"Yaşı:",yaş,"Boy:",boy)
Adı: Can
Yaşı: 16
Boy: 1.85
```

Değişkenler ve Değişken Tanımlama konusunda gördüğümüz örneği şimdi kullanıcıdan bilgi alarak yapalım.

- **yaş** değişkenine **yaş = int(input("Öğrencinin Yaşını Giriniz:"))** ifadesiyle kullanıcıdan int değeri,
- **boy** değişkenine **boy = float(input("Öğrencinin Boyunu Giriniz:"))** ifadesiyle kullanıcıdan float değeri,
- **adı** değişkenine **ad = input("Öğrencinin Adını Giriniz:"))** ifadesiyle kullanıcıdan str değeri, alarak atandı.

int ve **float** değerlerini kullanıcıdan alabilmek için **input()** fonksiyonu **int()** ve **float()** fonksiyonları içinde yazıldı.

```
>>> c = int(input("Değer Giriniz:"))
Değer Giriniz:can1993
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    c = int(input("Değer Giriniz:"))
ValueError: invalid literal for int() with base 10: 'can1993'
```

Eğer kullanıcıdan **int** değer girmesini beklerseniz ve kullanıcı **string** değer girerse değer hatası oluşacaktır. Kullanıcının hatalı giriş yapmasını engelleyen "Hatalar ve İstisnalar" konusunu ilerde göreceğiz.

3.4. Matematik Operatörleri

Matematik operatörleri; değişkenler ve veri tipleri üzerinde matematiksel işlemler yapmamızı sağlarlar. Basit Matematik işlemleri konusunda **toplama**, **çıkarma**, **çarpma** ve **bölme** operatörleri işlendi. Bu bölümde **üs alma**, **tam sayı bölmesi** ve **mod alma** operatörlerini göreceğiz.

Tablo 1 Matematik Operatörleri

Operatör	Bilgisayar Sembolü	Örnek	Sonuç
Toplama	+	6.7 + 2.1	8.8
Çıkarma	-	5.6 – 3.4	2.2
Çarpma	*	3 * 4	12
Bölme	/	40 / 8	5
Üs Alma	**	2 ** 3	8
Tam Sayı Bölme	//	22 // 7	3
Mod Alma	%	10 % 6	4

3.4.1. İşlem Önceliği

Matematiksel, mantıksal ve ilişkisel operatörlerin bir hiyerarşisi yani öncelikleri vardır. İşlemler, bu sıralamaya göre yapılmaz ise sonuç, beklendiği gibi çıkmayabilir. En içteki parantezden en dışakine doğru işlem yapılmalı, parantez içerisinde ise işlem önceliklerine dikkat edilmelidir. Aşağıdaki tabloda operatörlerin işlem önceliği yukarıdan aşağıya doğru listelenmiştir.

Tablo 2 Operatör Önceliği

İşlem Sırası	Açıklama
()	Parantez içerisindeki işlemler en içten en dışa doğru yapılır.
f(args...)	Fonksiyonlar
**	Kuvveti (Üs) operatörü
*, /, %, //	Çarpma, bölme, mod
+, -	Toplama, çıkarma

Kodlamadaki işlem önceliği hataları, yanlış sonuçlar alınmasına neden olabilir. Bu tür hatalar mantık hataları oldukları için uygulamada akışı engelleyen bir hata ile karşılaşılmaz. Bu nedenle mantık hatalarının nerede yapıldığının bulunması güçtür.

Tablo 3 İşlem Önceliği Örnekleri

Değişken	İşlem	İşlem	Çıktı
x = 10	x+y-z	10+15-20	5
y = 15	x-y*z	10-15*20	-290
z = 20	(x-y)*5	(10-15)*5	-25

3.4.2. İşlem Gruplama Operatörü Parantez ()

```
>>> x = 2
>>> y = 4
>>> z = 12
#Parantez ( ) kullanılarak işlem          #Parantez ( ) olmadan işlem
>>> (10+x+y) / (z+y) * (y+x) / x          >>> 10+x+y/z+y*y+x/x
3.0                                         29.333333333333336
```

Hem kodun okunabilirliğini arttırmak hem de işlemlerde hatayı engellemek için parantez kullanılır.

3.4.3. Üs Alma (**)

Bir sayının üssünü almak için ** operatörü kullanılır.

```
>>> 2 ** 6
64
>>> 2 ** 10
1024
>>> 3 ** 2
9
>>> 5 ** 3
125
>>> 7 ** 3
343
>>> a = 7
>>> a ** 2
49
>>> b = 6
>>> b ** 3
216
>>> x = 2
>>> y = 10
>>> x ** y
1024
```

Matematikte 2^6 şeklinde yapılan üs alma işlemi Python' da $2 ** 6$ şeklinde gösterilir.

```
>>> -2 ** 2
-4
>>> (-2) ** 2
4
>>> (-2) ** 3
-8
>>> a = -2
>>> a ** 2
4
>>> a ** 3
-8
```

Negatif sayılar ile üs alma işlemi yapıyorsanız işleminizin sonuçlarına dikkat etmeniz gerekir.

3.4.4. Karekök, Küp Kök Alma

```
>>> 4 ** (1/2)
2.0
>>> 16 ** (1/2)
4.0
>>> 9 ** (1/2)
3.0
>>> 4 ** 0.5
2.0
>>> 16 ** 0.5
4.0
>>> 9 ** 0.5
3.0
>>> 8 ** (1/3)
2.0
>>> 27 ** (1/3)
3.0
```

Matematikten bildiğimiz gibi, bir sayının karekökü, o sayının $(1/2)$ veya 0.5 üssüdür. Aynı mantıkla sayıların küp kökü $(1/3)$ üssüdür.

3.4.5. pow() Fonksiyonu İle Üs Alma

**	pow()	**	pow()
>>> 2 ** 6 64	>>> pow(2,6) 64	>>> (-2) ** 2 4	>>> pow(-2,2) 4
>>> 3 ** 2 9	>>> pow(3,2) 9	>>> (-2) ** 3 -8	>>> pow(-2,3) -8
>>> 5 ** 3 125	>>> pow(5,3) 125	>>> a = 2 8	>>> a = 2
>>> a = 7 49	>>> a = 7	>>> a ** 3 1024	>>> pow(a,3) 1024
>>> a ** 2 49	>>> pow(a,2) 49	>>> b ** 10 1024	>>> b = 10
>>> b = 6 216	>>> b = 6	>>> a ** b 1024	>>> pow(a,b) 1024
>>> b ** 3 216	>>> pow(b,3) 216	>>> 4 ** (1/2) 2.0	>>> pow(4,1/2) 2.0
>>> x = 2 1024	>>> x = 2	>>> 4 ** 0.5 2.0	>>> pow(4,0.5) 2.0
>>> y = 10 1024	>>> y = 10	>>> 8 ** (1/3) 2.0	>>> pow(8,1/3) 2.0
>>> x ** y 1024	>>> pow(x,y) 1024		

** operatörü ile yaptığımız her şeyi pow() fonksiyonuyla da yapılabilir.

3.4.6. Tam Sayı Bölme (//)

Bölünen	Bölen		Bölünen		Sonuç
13	3		13	=	4.333333
12	4		3		
=	Bölüm		Bölen		
1					
Kalan					

Üstteki şekil bölüm işleminde kullanılan elemanları göstermekte.

>>> 13 / 3 4.333333333333333	>>> a = 13
>>> 16 / 5 3.2	>>> b = 3
	>>> a / b 4.333333333333333

Bölme işleminde **Bölüm** sonucunun tam sayı kısmı veya nokta simgesinin solundaki değerdir.

13/3 işleminin sonucu 4.333333 sayısında **bölüm** 4'tür. 16/5 işleminde **bölüm** 3'tür.

>>> int(13 / 3) 4	>>> a = 13
>>> int(16 / 5) 3	>>> b = 3
	>>> int(a / b) 4

Bölme işlemi int() fonksiyonunun içinde yapılırsa sonuç **Bölüm** değerini verir.

>>> 13 // 3 4	>>> a = 13
>>> 16 // 5 3	>>> b = 3
	>>> a // b 4

Tam sayı bölme operatörü (//) bölme işlemindeki Bölüm değerini verir.

```

>>> 12 / 3
4.0 #float
>>> 12 // 3
4 #int
>>> 16 // 5
3 #int

```

```

>>> a = 12
>>> b = 3
>>> a / b
4.0 #float
>>> a // b
4 #int

```

İki tamsayının // operatörü ile yapılan bölüm işlemleri **int** (tam sayı) çıkar.

```

>>> # sonucu float çıkan tam sayı bölme işlemi
>>> 5 / 2
2.5 # float
>>> 5 // 2
2 # int

```

```

>>> 5 // 2.0
2.0 # float
>>> 5.0 // 2
2.0 # float

```

Eğer // operatörü ile biri float sayı işleme girerse sonuç float çıkar.

```

>>> # negatif sayılarla tam sayı bölme işlemi
>>> 5/2
2.5
>>> int(-5/2)
-2
>>> int(5/-2)
-2

```

```

>>> -5//2
-3
>>> 5//-2
-3
>>> -(5//2)
-2

```

// operatörü ile bölme işlemi yapıldığında sayılardan biri negatifse sonuç beklenildiği gibi çıkmaz. 5//-2 veya -5//2 işleminin sonucu -2 yerine -3 çıkar. Bu durumun oluşmaması için // işlemi **-(5//2)** gibi parantez içerisinde yapılabilir veya bölme işlemi **int(-5/2)** fonksiyonuyla yapılabilir.

3.4.7. Mod Alma Operatörü (%)

```

>>> 13 % 3
1
>>> 48 % 7
6

```

```

>>> a = 13
>>> b = 3
>>> a % b
1

```

Mod alma operatörü (%) bölme işlemindeki **Kalanı** verir.

3.4.8. İşaret Değiştirme (-)

```

>>> -6
-6
>>> a = 3
>>> -a
-3

```

```

>>> -(-2)
2
>>> b = -5
>>> -b
5

```

Kodunuzun içinde değişkenlerin veya sayıların işaretlerin değiştirmek için negatif operatörü “ - ” kullanılır.

3.5. Örnekler ve Alıştırmalar

3.5.1. Örnekler

3.5.1.1. IDLE Yardımıyla Klavyeden Girilen İki Tam Sayının Toplamını

Or_3_1_IDLEİkiSayıFarki

```
# girilen iki sayı ile yapılan işlemler
>>> birinciSayı = int(input("Birinci Sayıyı Girin: "))
Birinci Sayıyı Girin: 37
>>> ikinciSayı = int(input("İkinci Sayıyı Girin: "))
İkinci Sayıyı Girin: 43
>>> toplam = birinciSayı + ikinciSayı
>>> fark = birinciSayı - ikinciSayı
>>> çarpım = birinciSayı*ikinciSayı
>>> bölme = birinciSayı/ikinciSayı
>>> ortalama = toplam/2
>>> print("Toplam :",toplam)
Toplam : 80
>>> print("Fark :",fark)
Fark : -6
>>> print("Çarpım :",çarpım)
Çarpım : 1591
>>> print("Bölme :",bölme)
Bölme : 0.8604651162790697
>>> print("Ortalama :",ortalama)
Ortalama : 40.0
```

Bu alıştırmada kullanıcı klavyeden sırasıyla iki değer alarak alınan değerlerle işlem yapan bir uygulama yazdık. IDLE ile yaptığınız çalışmalarda her kod bir komutmuş gibi girilir. IDLE bize kod yazmak için gerçekçi bir kullanım sunmaz sadece çok kısa denemeleri yapma imkânı sunar.

3.5.1.2. Pycharm Yardımıyla Klavyeden Girilen İki Tam Sayının Farkı

Or_3_2_IkiSayıFarki.py

```
# Girilen iki sayının farkını alma
print("Girilen İki Sayının Farkı(a - b)")
birinciSayı = int(input("Birinci Sayıyı Girin:"))
ikinciSayı = int(input("İkinci Sayıyı Girin:"))

fark = birinciSayı - ikinciSayı

print("Fark :", fark)

# Ekran çıktısı
"""
Birinci Sayıyı Girin:33
İkinci Sayıyı Girin:13
Fark : 20
"""
```

Pycharm kod yazmak için özelleşmiş bir editör programıdır. Bu tip programlara *ide(Integrated Development Environment)* denir. Yazdığımız kodlar “.py” uzantılı dosyalarda saklanır. Pycharm Python kod dosyasını oluşturma ve bu dosyayı çalıştırmak için tüm araçları bize sunar. Pycharm üzerinde hazırlanan kod dosyası çalıştırıldığında programımız çalışacak ve sizden iki tam sayı değeri alıp farklarını hesaplayarak size sunacaktır.

3.5.1.3. Kişilerin Yaşı

Or_3_3_KisilerinYasi.py

```
# Kişilerin yaşını bulma
print("Belli Bir Yıla Göre Kişinin Yaşını Hesaplama")
dogumYılı = int(input("Doğum Yılı Girin:"))
hesapYılı = int(input("Hesaplama Yılı Girin:"))

yaş = hesapYılı - dogumYılı

print(yaş, " yaşında")
# Ekran çıktısı
"""
Doğum Yılı Girin:1881
Hesaplama Yılı Girin:1923
42 yaşında
"""
```

Bir kişinin doğum yılını biliyorsanız yaşını hesaplamak için bir program kullanabilirsiniz. Üstte yazılan kodu **Mustafa Kemal ATATÜRK** 'ün Cumhuriyet ilan edildiğinde kaç yaşında olduğunu bulmak için kullandık.

3.5.1.4. Çemberin Çevresi ve Alanı

Or_3_4_CemberinCevresiAlani.py

```
# Çemberin Çevresi ve Alanı
print("Yarı Çapı Girilen Çemberin Çevresini ve Alanını Bulma")
r = float(input("Çemberin Yarı Çapını Girin:"))
pi = 3.14159

print("Çemberin Çevresi:", 2*pi*r)
print("Çemberin Alanı : ", pi*(r**2))

# Ekran çıktısı
"""
Çemberin Yarı Çapını Girin:4.12
Çemberin Çevresi: 25.8867016
Çemberin Alanı : 53.326605296
"""
```

Çemberin yarı çapı girilerek çevresi ve alanı bulunabilir. Bu kodda çemberin yarı çapı kullanıcıdan **float** olarak alınmış.

3.5.1.5. Girilen İki Sayının Ortalaması

Or_3_5_GirilenIkiSayininOrtalamasi.py

```
# Kullanıcı tarafından girilen iki sayının ortalaması
a = float(input("Birinci Sayıyı Giriniz: "))
b = float(input("İkinci Sayıyı Giriniz : "))

print("İki Sayının Ortalaması:", (a + b) / 2)
# Ekran çıktısı
"""
Birinci Sayıyı Giriniz: 10.4
İkinci Sayıyı Giriniz : 30.6
İki Sayının Ortalaması: 20.5 """
```

Bu örnekte kullanıcıdan alınan değerler **a** ve **b** değişkenlerine atandı. Ortalama hesabı **print** fonksiyonun içerisinde ekrana yazdırılırken **(a + b) / 2** işlemi yapılarak yazdırıldı.

3.5.2. Alıştırmalar

3.5.2.1. IDLE Kullanılarak Yapılan Hesaplamalar

Aşağıdaki işlemlerin sonuçlarını kendiniz hesaplayın ve aynı işlemi etkileşimli kabukta(IDLE) deneyin.

AI_3_1_IDLEKullanılarakYapılanHesaplamalar.py

- | | | |
|-----------------|---------------|------------------------------|
| 1. $3+5$ | 4. $15.3+4.7$ | 7. $5-2*(7-12)+6/3$ |
| 2. $17-2*4-5+3$ | 5. $4*(1/2)$ | 8. $5*3-7*2+(6/3)-2*4*(0.5)$ |
| 3. $18.5-3.2$ | 6. $18*(0.5)$ | 9. $((3+5)*4-(9-5)*3)/5$ |

x = 3 için alttaki her işlemin sonucunda x değişkeninin son değerini IDLE kullanarak bulun.

- | | | |
|------------------|------------------------|----------------------------|
| 1. $x = x + 1$ | 4. $x = 2*x + 7$ | 7. $x += 1$ |
| 2. $x = x - 1$ | 5. $x = (x + 9)/3 + 4$ | 8. $x *= 5$ |
| 3. $x = x*3 + 2$ | 6. $x = x + 9/3 + 4$ | 9. $x = ((7 + x)/2)*4 + x$ |

Altta yazan işlemleri IDLE kullanarak çözün.

- | | | |
|--|--|--|
| 1.
$t = 6$
$y = 9$
$t = t + y$
$t = ?$ | 2.
$c = 6$
$a = c$
$t = 5$
$d = 4$
$t = d + a + t$
$t = ?$ | 3.
$a = 3$
$b = 2$
$c = 4$
$t = 1$
$t = a*b - c + t + 5$
$t = ?$ |
|--|--|--|

3.5.2.2. Öğrenci Bilgileri

AI_3_2_OgrenciBilgileri.py

```
# Ekran çıktısı
"""
Öğrenci Bilgileri Giriş Ekranı
Öğrenci Adı:Ahmet
Öğrenci Soyadı:TUNA
Baba Adı:Mehmet
Anne Adı:Ayşe
Sınıfı:9-B
Doğum Yılı:2003

Öğrenci Bilgileri
Öğrencinin Adı: Ahmet
Öğrencinin Soyadı: TUNA
Baba Adı: Mehmet
Anne Adı: Ayşe
Öğrencinin Doğum Yılı: 2003
Öğrencinin Sınıfı: 9-B
Öğrencinin Yaşı: 14
"""
```

Öğrencinin Adı, Soyadı, Baba Adı, Anne Adı, Sınıfı ve Doğum Yılı bilgilerini alan ve bilgileri ekrana yazdıran programı yazınız. Ekran çıktısı üstteki gibi olabilir. Doğum yılı kullanıcıdan alınarak öğrencinin yaşı hesaplanarak yazdırılacak.

3.5.2.3. Vücut Kitle Endeksi(BMI) Hesaplama

AI_3_3_VucutKitleEndeksi

```
# Ekran çıktısı
"""
Vücut Kitle Endeksi (BMI) Hesaplama
Boyunuzu Metre Cinsinden Giriniz (1.85 gibi) :1.75
Kilonuzu Tam Sayı Olarak Giriniz (85 gibi) :80
Vücut Kitle Endeksiniz (BMI) : 26.122448979591837
"""
```

Kullanıcının girdiği boy ve kilo değerlerine göre vücut kitle endeksi bulan programı yazın.

Vücut Kitle Endeksi = $\frac{\text{kilo}}{\text{boy}^2}$ Formülünü kullanarak programınızı yazın. **boy**² için üs alma operatörünü kullanın.

3.5.2.4. Araç Yakıt Tüketimi

AI_3_4_AracYakitTuketimi

```
# Ekran çıktısı
"""
Araç Kullanım Bilgileri Giriş Ekranı
Gidilen Yol (km) :428.8
Aldığınız Yakıt Miktarı (lt) :27.982
Alınan Yakıt Tutarı:146.6
100km 'de Harcanan Yakıtın Litresi: 6.525652985074625
100km 'de Harcanan Yakıtın Tutarı: 34.18843283582089
"""
```

Eğer yakıt deposu tam dolu bir araçla yola çıkarsanız gideceğiniz yere vardığınızda aracın deposunu yeniden doldurduğunuzda aracın yakıt tüketim değerlerini hesaplayabilirsiniz. Alttaki Formülleri kodunuzda kullanarak araç yakıt tüketim hesabı yapan bir program yazın.

Yüz Kilometrede Harcanan Litre Benzin = $\frac{\text{Alınan Yakıt Litre}}{\text{Gidilen Yol Kilometre}} \cdot 100$

Yüz Kilometrede Harcanan Benzin Tutarı = $\frac{\text{Alınan Yakıt Tutarı}}{\text{Gidilen Yol Kilometre}} \cdot 100$

3.5.2.5. İki Değişkenin Değerini Birbiriyle Değiştirme

AI_3_5_IkiDegiskeninDegerleriniDegistirme.py

```
# Ekran çıktısı
"""
İkinci Değişkenin Değeri Birinciden Büyük Olmalı
Birinci Değişkeni Girin: 30
İkinci Değişkeni Girin: 45
Değişkenler İlk Durumuda Yapılan Çırkama İşleminin Sonucu: -15
Değişkenlerin Değerleri Bir Biriyle Değiştirilince Sonuç: 15
"""
```

Kullanıcıdan iki pozitif tam sayı değeri aldırın, ikinci girilen sayı ilk girilen sayıdan büyük olsun. İlk önce birinci değişkenden ikinci değişkeni çıkararak sonucu yazdırın. Daha sonra değişkenlerin değerlerini birbiriyle değiştirerek aynı işlemi yeniden yaptırıp sonucu yazdırın.

3.5.2.6. Sonucu Tam Sayı Çıkan Bölme İşlemi

AI_3_6_TipDonusumu.py

```
# Ekran çıktısı
"""
Sonucu Tam Sayı Çıkan Bölme İşlemleri İçin Değer Girin
Bölünecek Sayıyı Girin: 36
Bölecek Sayıyı Giriniz: 12
Tip Dönüşümü Yapılmadan Önce Sonuç: 3.0
Tip Dönüşümü Yapıldıktan Sonra Sonuç: 3
Tam Sayı Bölme Operatörü İle Sonuç: 3
"""
```

Python’ da $36/12$ gibi bölme işleminin sonucu tam sayı olsa bile sonuç her zaman 3.0 gibi float çıkar. Yazacağınız programda sonucu tam sayı çıkacak iki sayıyı birbiriyle bölün. Önce sonucu olduğu gibi yazdırın sonra sonucun tipini tam sayıya çevirip yeniden yazdırın son olarak tam sayı bölme “//” operatörünü kullanarak bir daha hesaplayın. $36/12$, $16/4$, $27/9$ sayılarını kullanabilirsiniz.

3.5.2.7. Önce İşlem Sonra Atama

AI_3_7_OncelslemSonraAtama.py

```
# Ekran çıktısı
"""
Klavyeden 1-20 Arası Pozitif Tam Sayı Girin: 12
Girilen sayının değeri 2 katının 5 fazlası kadar arttırıldı
Sayının Yeni Değeri : 29
"""
```

Kullanıcının 1 – 20 arası gireceği pozitif bir tamsayının değerini iki katının beş fazlası olacak şekilde arttırıp yeni değeri ekrana yazdıran programı yazın. Değişkenin değerini önce işlem sonra atama kuralıyla arttırmayı unutmayın.

3.5.2.8. İkinci Dereceden Bir Bilinmeyenli Denklemlerin Kökleri

AI_3_8_IkinciDerecedenDenklem.py

```
# Ekran çıktısı
"""
Tam Sayı a: 1
Tam Sayı b: -4
Tam Sayı c: 4

Birinci Kök: 2.0
İkinci Kök: 2.0
"""
```

$a, b, c \in \mathbb{R}, a \neq 0$ ve x bilinmeyen olmak üzere $ax^2 + bx + c$ şeklinde ifade edilen ikinci dereceden bir bilinmeyenli denklemlerin köklerini $a \neq 0, b \neq 0, c \neq 0$ için aşağıdaki formülleri kullanarak bulan programı yapın.

$$\Delta = b^2 - 4ac$$

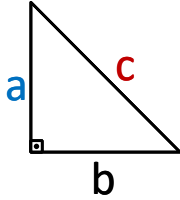
$$x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$$

3.5.2.9. Hipotenüs Uzunluğu Bulan Program

Al_3_9_Hipotenüs.py

```
# Ekran çıktısı
"""
a Kenarının Uzunluğu: 3
b Kenarının Uzunluğu: 4

c Kenarının Uzunluğu (Hipotenüs): 5.0
"""
```



$$a^2 + b^2 = c^2$$

Kullanıcıdan dik üçgenin dik kenarları olan a ve b kenarlarının uzunluklarını alan ve hipotenüs uzunluğu olan c kenarını bulan programı yazın.

3.5.2.10. Girilen Saniye Değerini Saat Dakika Saniye Olarak Gösteren Program

Al_3_10_SaniyeyiSaatDakikaSaniyeyeAyirma

```
# Ekran çıktısı
"""
Saniye: 29000
8 : 3 : 20
"""
```

Kullanıcıdan saniye bilgisini alan saniyeyi Saat : Dakika : Saniye olarak gösteren programı yazınız.