

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 programlama dilleri 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 ve ondalıksız 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 matematikte olduğu gibi iki tam sayının birbirine oranı ile ifade edilebilen sayılardır. Ondalık 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

<pre>>>> # Toplama >>> 3 + 5 8 → int</pre>	<pre>>>> #Çıkarma >>> 7-12 -5 → int</pre>	<pre>>>> #Bölme >>> 5/2 2.5 → float</pre>
<pre>>>> #Çıkarma >>> 13-5 8 → int</pre>	<pre>>>> #Çarpma >>> 17 * 3 51 → int</pre>	<pre>>>> #Bölme >>> 6/2 3.0 → float</pre>

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 karakterler (**string**) ifadesi kısaca **str** olarak kullanılır. Örnek: "Merhaba", "Dünya", "12322"

<pre>>>> #Tek tırnak ile >>> 'Tuna POTUR' 'Tuna POTUR'</pre>	<pre>>>> #Çift Tırnak ile >>> "Tuna POTUR" 'Tuna POTUR'</pre>	<pre>>>> #Üç Tırnak ile >>> """Tuna POTUR""" 'Tuna POTUR'</pre>
<pre>>>> """üç tırnak bir den çok satırlı metinler için kullanılır"""</pre>		

3.1.5. Alıştırmalar

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

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

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. Diğer programlama dillerinin aksine Python’ da değişkenleri önceden tanımlamanız gerekmez. Bir değişkene bir değer atamak için atama ifadesi olan eşittir (=) operatörü kullanılır.

```
>>> # 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 değişkenlere hangi tipte değer verildiyse değişken o tipte oluşturulmuştur.

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

print()

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

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

"\n" bir satır alta geçmek için kullanıldı.

"Adınız:", "Yaşınız:" ve "Boyunuz" karakter verileri doğrudan ekrana yazdırılmak için fonksiyona girildi.

adı, yaş ve boy değişkenleri içinde tutulan değerlerin ekrana bastırılması için fonksiyona girildi.

3.2.1. Değişken Tiplerinin Öğrenilmesi ve Değişken Tipinin Kod Akışı İçerisinde Değiştirilmesi

type()

Değişkenlerin tipleri, o anda hangi değeri taşıyorsa o değerın tipi ile aynıdır. Değişkenlerin tiplerini öğrenmek için type() fonksiyonu kullanılır. type(değişken) şeklinde çalıştırılan type() fonksiyonu bize değişkenin hangi tipte olduğunu döner.

```
>>> 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.*

```

>>> 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.

Değişkenin program çalıştığı sürece belli bir anlamı ve rolü olduğu için tipi genellikle değişmez. Bu yüzden değişkenleri oluşturduğumuz tiple kullanmalıyız. Çok nadir durumlar dışında değişkenlerin tipini değiştirmekten kaçınmalıyız.

3.2.2. 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 başlamaz. Değişken isimleri bir karakter ile başlamalıdır.
4. Python3 de değişken adı olarak kullanabileceğimiz karakterler:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzÇĞİÖŞÜçğİöşü_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'

```

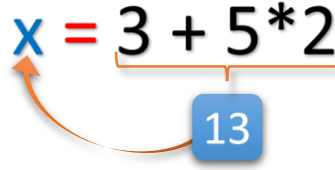
TELEFON = “417 00 01” ve telefon = “415 11 12” iki ayrı değişkenlerdir.

Hatalı Değişken İsimleri Örnekleri

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

3.2.3. 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 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.

3.2.4. Değişkenlerin Değerini Arttırma

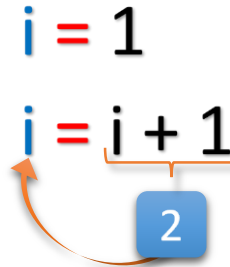
```
>>> #Önce işlem sonra atama
>>> x = 5
>>> x = x*2 + 3
>>> print("X değeri :",x)
X değeri : 13
```

Burada sizce bir matematik kuralını mı ihlal ettik? Sorusu akla gelebilir. Eşittir operatörünün kullanımını düşünersek önce sağ taraftaki işlem yapılacak sonra sonuç x değişkenine atanacaktır.

```
>>> #Önce işlem sonra atama
>>> #x' in değerini 1 arttırma
>>> x = 13
>>> x = x + 1
>>> print("X değeri :",x)
X değeri : 14
```

$x = x + 1$ işlemiyle x değişkenini değeri 1 artar. Değişkenlerin değerlerini kendisiyle 1'i toplayarak arttırmak ileride göreceğimiz döngüler konusunda çok sık kullanılan bir yöntemdir.

```
>>> i = 1
>>> i
1
>>> i = i + 1
>>> i
2
>>> i = i + 1
>>> i
3
```



Bu kod örneğinde görüldüğü gibi $i = i + 1$ işlemi yapmak değişkenin değerini 1 arttırılıyor.

```
>>> a = 1
>>> a
1
>>> a+=1
>>> a
2
>>> a+=1
>>> a
3
```



$i = i + 1$ yerine Python 'da aynı işlemi yapmak için $i += 1$ ifadesi kullanılır. $i += 1$ ile $i = i + 1$ aynı işi yapar fakat $i += 1$ yazmak daha anlaşılır ve okunabilir olduğu için tercih edilir. $i += 1$ işleminde $+=$ operatörü i değişkeninin artırılacağını, 1 de i değişkenin değerinin birer artırılacağını belirtiyor.

```
>>> b = 6
>>> b-=2
>>> b
4
>>> b-=2
>>> b
2
```

Aynı mantıkla değişkenlerin değerleri azaltılabilir. **b -= 2** işlemi **b** değerini **2** azaltır.

```
>>> c = 4
>>> c*=3
>>> c
12
>>> c*=3
>>> c
36
```

c*=3 işlemi **c** değeri **3** ile çarparak artırır.

3.2.5. İki Değişkenin Değerini Birbiriyle Değiştirme

```
>>> a = 3
>>> b = 5
```

a ve **b** değişkenlerinde bulunan değerleri bir biriyle yer değiştirmek gerekebilir.

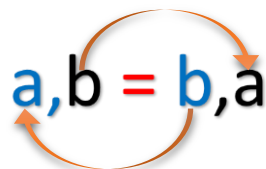
```
>>> a = b
>>> a
5
>>> b
5
```

a = b gibi bir eşitlik yaparsak **b** değişkeninde bulunan değer **a** değişkenine geçecektir ve **a** değişkeni **b** değişkeni ile aynı değere sahip olacaktır.

```
>>> a = 3
>>> b = 5
>>> takas = a
>>> a = b
>>> b = takas
>>> 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
>>> 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.

3.2.6. Aynı Değere Sahip Değişkenler Tanımlama

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

Yazdığınız 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 değişkenlere tanımlamak için aynı değere sahip değişkenleri üstte görüldüğü gibi kolay bir şekilde tanımlayabilirsiniz.

3.2.7. Tip Dönüşümleri

3.2.7.1. Otomatik Tip Dönüşümleri

İki aynı sayı tipiyle bölme hariç işlem yapıldığında işlemin sonucu işlemin yapıldığı veri tipinde olacaktır.

```
#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 verinin 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'>
```

float tipinde iki değişkenin toplamı sonucu tam sayı olsa bile sonuç yine **float** tipinde olacaktır.

Görüldüğü gibi **float** ve **int** tipinde veriler toplandığında sonuç her zaman **float** çıkar. Aslında programlama dillerinde iki farklı tipte veri birbiriyle işleme giremez. Fakat birçok programlama dilinde olduğu gibi Python’ da tip dönüşümünü otomatik olarak yapar. Python **int** sayı tipindeki veriyi arka planda **float** sayı tipine çevirir ve işlemi **float** sayılar arasında yapar. Özetle **float** ve **int** sayı tipleriyle işlem yapıldığında **int** sayı tipleri **float** sayı tipine çevrildiği için sonuç 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 sonuç her zaman **float** çıkar. Bunun sebebi bölme işlemlerinin sonuçlarının çoğunlukla **float** çıkmasından kaynaklanmaktadır.

3.2.7.2. İsteğe Bağlı Tip Dönüşümleri

İşlemler içerisinde otomatik tip dönüşümleri olduğu gibi isteğe bağlı olarak tip dönüşümü yapılabilir.

int(), float(), str()

Bu fonksiyonlar değer olarak aldıkları verileri kendi tiplerine dönüştürür.

int(3.14) → 3 tam sayı

float(5) → 5.0 ondalıklı sayı

str(12) → '12' karakter verisi

```
>>> # ondalıklı sayıyı tam sayıya çevirme
>>> int(3.14)
3
>>> pi = 3.14
>>> int(pi)
3
```

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


```
>>> # tamsayıyı ondalıklı sayıya çevirme
>>> float(7)
7.0
>>> float(-9)
-9.0
>>> a = 53
>>> float(a)
53.0
```

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.

```
>>> # tam ve ondalıklı sayıyı string' e çevirme
>>> str(47)
'47'
>>> a = 63
>>> str(a)
'63'
>>> str(3.141592)
'3.141592'
>>> pi = 3.141592
>>> str(pi)
'3.141592'
```

Sayıları karaktere çevirmek için `str()` fonksiyonu kullanılır (`str`, `String`' in kısaltmasıdır). `str()` fonksiyonuyla sayıyı oluşturan tüm rakamlar veya nokta birer karaktere dönüşecektir.

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

Bir string'i, tamsayıya çevirmek için `int()` fonksiyonu, 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'
>>> int("1981") # tip dönüşümü hatasının düzeltilmesi
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'
>>> float("3.14324324") # tip dönüşümü hatasının düzeltilmesi
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")
88
>>> int("28") + 37
65
>>> 12 + float("2.17")
14.17
>>> 19.23 + float("-5.5")
13.73
```

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

```

>>> # 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
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.

input()

Fonksiyonu kullanıcıdan değer almak için kullanılır. Kullanıcıya mesaj vermek için **input** fonksiyonuna değer girilir. **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:85
>>> 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:")

```

İşlemleriyle 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 alır. 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** fonksiyonun içinde kullanmanız gerekir. Böylelikle kullanıcıdan aldığınız **string** değer **int** fonksiyonuyla tam sayıya dönüştürülür. Tam sayıya dönüştürülen kullanıcı verisi tam sayılarla 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.

```
>>> 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.

3.2.8. Örnekler

3.2.8.1. IDLE Yardımıyla Klavyeden Girilen İki Tam Sayının Toplamını

```
# 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 olarak 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.

3.2.8.2. Pycharm Yardımıyla Klavyeden Girilen İki Tam Sayının Farkı

Pycharm kod yazmak için özelleşmiş bir editör programıdır. Bu tip programlara *ide(Integrated Development Environment)*denir. Yazdığımız kodlar “.py” uzantılı dosyalarda saklanır. Kodları içeren Python dosyası çalıştırıldığında yazdığınız program da çalışacaktır. Pycharm Python kod dosyasını oluşturma ve bu dosyayı çalıştırmak için tüm araçları bize sunar.

3_2_8_2_IkiSayiFarki.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
"""
```

İki sayının farkı uygulamasında kodlarımız “3_2_8_2_IkiSayiFarki.py” dosyasında saklanır. Pycharm üzerinde hazırlanan kod dosyası çalıştırıldığında programımız çalışacak ve sizden iki tam sayı değeri alıp farklarını hesaplayarak size sunacaktır.

3.2.8.3. Kişilerin Yaşı

3_2_8_3_KisilerinYasi.py

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

yaş = hesapYılı - doğumYılı

print(yaş, " yaşımda")
```

```
# Ekran çıktısı
"""
Doğum Yılını Girin:1881
Hesaplama Yılını 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.

3.2.8.4. Çemberin Çevresi ve Alanı

3_2_8_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*r)

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

Çemberin yarı çapı biliniyorsa çevresi ve alanı bulunabilir. Çemberin alanını ve çevresini bulma kodunda çemberin yarı çapı kullanıcıdan **float** olarak alınmış.

3.2.8.5. Girilen İki Sayının Ortalaması

3_2_8_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** fonksiyonunun içerisinde ekrana yazdırılırken **(a + b) / 2** işlemi yapılarak yazdırıldı.

3.2.9. Alıştırımlar

3.2.9.1. IDLE Değişken Kullanılarak Yapılan Hesaplamalar

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

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

3.2.9.2. Öğrenci Bilgileri

3_2_9_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.

3.2.9.3. Vücut Kitle Endeksi(BMI) Hesaplama

3_2_9_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 = $\text{kilo}/(\text{boy}^2)$ formülünü kullanarak programınızı yazın.

3.2.9.4. Araç Yakıt Tüketimi

3_2_9_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 bir aracın yakıt deposunu doldurup yola çıkarsanız gideceğiniz yere vardığınızda aracın deposunu yeniden doldurursanız aracın yakıt tüketim değerlerini hesaplayabilirsiniz.

Yüz Kilometrede Harcanan Benzin Litresi = $(\text{Alınan Yakıt Litre} / \text{Gidilen Yol Kilometre}) * 100$

Yüz Kilometrede Harcanan Benzin Tutarı = $(\text{Alınan Yakıt Tutarı} / \text{Gidilen Yol}) * 100$

Formüllerini kullanarak programınızı yazın.

3.2.9.5. İki Değişkenin Değerini Birbiriyle Değiştirme

3_2_9_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.

3.2.9.6. Sonucu Tam Sayı Çıkan Bölme İşlemi

3_2_9_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
"""
```

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. $36/12$, $16/4$, $27/9$ gibi sayılar kullanabilirsiniz.

3.2.9.7. Önce İşlem Sonra Atama

3_2_9_7_OncelIslemSonraAtama.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.