Değer ve Tür

Değer: programın işledikleri: girdi, ara işlem, çıktı.

```
>>> print 1 + 3
4
```

→ buradaki 1 ve 3, çıktıdaki 4 değerdir

```
>>> print "merhaba dunya!"
merhaba dunya !
```

→ buradaki “merhaba dunya!” değerdir.

Tür

→ bu değerlere farklı türe sahiptir

→ 1 - tamsayı (integer) iken “merhaba dunya!” - karakter dizisidir (dizgi, string)

→ sayı ve dizgiyi ayırt eden şey: çift tırnak

→ dolayısıyla print “4” ile print 4’deki dörtler farklı türdedir

```
>>> print "23" + "45"
2345
>>> print 23 + 45
68
```

→ burada söylenecek çok şey: + işlecinin dizgi ve sayılarda farklı kullanımı

type

→ değerin türünü öğrenmede type komutu kullanılır

```
>>> type("merhaba dunya!")
<type 'str'>
>>> type(4)
<type 'int'>
```

Diğer türler

→ tamsayı ve dizgi dışında da türler vardır

→ gerçel sayılar için: float. ör. 3.2

```
>>> type(3.2)
<type 'float'>
```

→ type("3.2") ve type("17") ne üretir?

tek - çift - üç tırnak

→ dizgiler tek-' ve çift-" tırnakla yazılır

→ dizgi içerisinde tek veya çift tırnağa gereksinim duyulması

```
>>> print 'OMU'nun gelecegi'
  File "<stdin>", line 1
    print 'OMU'nun gelecegi'
              ^
SyntaxError: invalid syntax
```

print: büyük sayıları göstermek

→ binler ayracı olarak , kullanmak isteyebilirsiniz

```
1  >>> print 1, 000, 000
2  1 0 0
3  >>>
```

→ python bunu üç elemanlı **liste** olarak yorumlar

→ gerçel sayılar için .–nokta kullan

Değişkenler: kap

- → kullanıcıdan aldığımız şeyleri
- → tuttuğumuz kablar
- → program içerisinde hesapladığımız şeyleri tuttuğumuz kablar
- → programlama jargonunda bu kaplara $\Longrightarrow$ **değişken***denilir*

Değişkenler

→ değişken, isminden de anlaşılacağı üzere değeri değiştirilebilir

→ **atama** cümlesi yeni değişken yaratır ve değerini atar

```
1  >>> mesaj = "merhaba, dunya!"
2  >>> n = 17
3  >>> pi = 3.14159
```

Değişkenler

→ değişken, isminden de anlaşılacağı üzere değeri değiştirilebilir

→ **atama** cümlesi yeni değişken yaratır ve değerini atar

```
1  >>> mesaj = "merhaba, dunya!"
2  >>> n = 17
3  >>> pi = 3.14159
```

→ üç atama, üç değişken: mesaj, n, pi

→ üç tür: dizgi, tamsayı, gerçel

değişkenler

- → değişkenin ismi, diğerlerinden ayırt etmek
- → değişkenin değeri, içerisinde tuttuğu
- → **değişkenin türü, içerdiği tür**
  - → veri yapıları
- → türü atama sırasında dinamik belirlenir

atama işleci

- → atama işleci, =-eşittir
- → sol taraf= sağ taraf
- → genel kural: sağdakini yorumla, sola ata

```
>>> 17 = n
  File "<stdin>", line 1
SyntaxError: can't assign to literal
```

print + değişkenler

→ print cümleleri değişkenlerle de kullanılır

```
1  >>> print mesaj
2  merhaba, dunya !
3  >>> print n
4  17
5  >>> print pi
6  3.14159
```

→ değişkenin (ör. n) o anki değeri (ör. 17) ekrana yazılır

print + değişkenler: karmaşık cümleler

→ print cümlesinde hem mesaj hem de değer yazdırabiliriz

```
>>> a = 5
>>> b = 3
>>> print "a =", a, "b =", b
a = 5 b = 3
>>> print "a =", a, "\nb =", b
a = 5
b = 3
```

print + değişkenler: karmaşık cümleler

→ aritmetik işlem sonucunu yazdırabiliriz

```
>>> print "a ile b'yi carparsak", a * b, "elde ederiz"
a ile b'yi carparsak 15 elde ederiz
```

→ veya

```
>>> print a, "ile", b, "'yi carparsak", a * b, "elde ederiz"
5 ile 3 'yi carparsak 15 elde ederiz
```

print + değişkenler: karmaşık cümleler: formatlama

→ mesaj içerisinde özel işaretler (%s) yardımıyla dizgi içine gömmekte mümkündür

```
1  >>> print "%s ile %s'yi carparsak %s elde ederiz" % (a, b, a*b)
2  5 ile 3'yi carparsak 15 elde ederiz
```

print + değişkenler: karmaşık cümleler: formatlama

→ mesaj içerisinde özel işaretler (%s) yardımıyla dizgi içine gömmekte mümkündür

```
>>> print "%s ile %s'yi carparsak %s elde ederiz" % (a, b, a*b)
5 ile 3'yi carparsak 15 elde ederiz
```

→ daha ileri düzey formatlama yapıları vardır

```
>>> print "{0} ile {1}'yi carparsak {2} elde ederiz".format(a, b, a*b)
5 ile 3'yi carparsak 15 elde ederiz
```

→ şunlara da bakın: repr, rjust

değişkenler + tür

$\longrightarrow$ değişkenlerin de türü vardır

değişkenler + tür

- → değişkenlerin de türü vardır
- → içeriği neyse türü odur
- → ör. mesaj, içeriği dizgi olduğundan türü de dizgidir
- → çalışma zamanında öğrenmenin yolu type işlevini kullanmaktır

```
>>> type(mesaj)
<type 'str'>
>>> type(n)
<type 'int'>
>>> type(pi)
<type 'float'>
```

türlerle çalışmak

→ daha ileri düzey işler yapmak gerekebilir

```
1  from types import *
2  def delete(mylist, item):
3
4      if type(item) is IntType:
5              del mylist[item]
6      else:
7              mylist.remove(item)
```

türlerle çalışmak

→ daha ileri düzey işler yapmak gerekebilir

```
1  from types import *
2  def delete(mylist, item):
3
4      if type(item) is IntType:
5              del mylist[item]
6      else:
7              mylist.remove(item)
```

→ dikkat: if type(item) is IntType:

→ IntType, FloatType, StringType

→ işlevler (def): 3. hafta

→ koşul ifadeleri (if, else): 4. hafta

→ listeler: 9 hafta

→ modüller (from, import): 10. hafta

değişkenler: isim ve anahtar kelimeler

- → anlamlı isimler
- → bu değişkeni ne için kullandığını belgelemenin kısayolu
- → uzunluğu isteğe bağlıdır
- → harf + rakam içerebilir

değişkenler: isim ve anahtar kelimeler

- → anlamlı isimler
- → bu değişkeni ne için kullandığını belgelemenin kısayolu
- → uzunluğu isteğe bağlıdır
- → harf + rakam içerebilir **mutlaka** harfle başlar

değişkenler: isim ve anahtar kelimeler

- → anlamlı isimler
- → bu değişkeni ne için kullandığını belgelemenin kısayolu
- → uzunluğu isteğe bağlıdır
- → harf + rakam içerebilir **mutlaka** harfle başlar
- → BÜYÜK - küçük harf duyarlıdır
- → büyük harfler genelde özel değerleri tutmada kullanılır
- → alt çizgi-_ de kullanılabilir

değişkenler: isimler: stiller

- → **b**: tek küçük harf, **B**: tek büyük harf
- → **küçükharf**, **BÜYÜKHARF**
- → **altçizgi_ile_küçükharf**, **ALÇİZGİ_İLE_BÜYÜKHARF**
- → **BaşHarfiBüyükHarf**
- → **karışıkBüyükKüçükHarf**
- → **Altçizgi_İle_İlk_Harfleri_Büyük_Kelimeler** (iğrenç)
- → isimlendirme kuralı: 'PEP 8 Style Guide for Python Code'

değişkenler: isimler: stiller

- → var olan çalışmanın stiline uyun
- → kendi stilinizde tutarlı olun
- → altçizgi ile başlayan ve biten değişken isimlendirmelerinde dikkatli olun Python'la çakışabilir
- → bunlar birbirine karışır: **l - 1 - I** ve **O - 0**, uzak durun
- → sabit değerleri içeren değişkenler **ALÇİZGİ_İLE_BÜYÜKHARF** stilinde olmalı

değişkenler: isimler: örnek

→ geçerli

```
>>> ad = "piton"
>>> versiyon = 2.6
>>> yil = 2010
```

değişkenler: isimler: örnek

→ geçerli

```
>>> ad = "piton"
>>> versiyon = 2.6
>>> yil = 2010
```

→ geçersiz

```
>>> 300sipartali = "Zack Snyder"
SyntaxError: invalid syntax
>>> more $ = 10000
SyntaxError: invalid syntax
>>> class = " Bilgisayar Programlama 1"
SyntaxError: invalid syntax
```

Açıklama

- → 300sipartali: sayıyla başlamamalı
- → more$: özel karakter (ör. $) içermemeli
- → class: anahtar kelimeler olamaz

anahtar kelimeler

- anahtar (veya rezerve edilmiş) kelimeler: dile özgü, dilin kural ve yapısını tanımlar
- değişken isimlerinde **kullanılamazlar**

`and`	`del`	`for`	`is`	`raise`
`assert`	`elif`	`from`	`lambda`	`return`
`break`	`else`	`global`	`not`	`try`
`class`	`except`	`if`	`or`	`while`
`continue`	`exec`	`import`	`pass`	`yield`
`def`	`finally`	`in`	`print`	

değişkenlerle çalışma

→ değer ata ve aritmetik işlem yap

```
>>> x = 6
>>> y = 5
>>> x + y
11
```

değişkenlerle çalışma

→ değer ata ve aritmetik işlem yap

```
>>> x = 6
>>> y = 5
>>> x + y
11
```

→ aynı anda çoklu atama

→ değerlerin üzerine yazma

```
>>> a = b = c = 10
>>> a + b + c
30
>>> a = b = c = 20
>>> a + b + c
60
```

büyük değerlerle çalışma

→ büyük değerlerle çalışma

```
>>> fmin = 0.000000000033333
>>> fmax = 3333300000000000.0
>>> fmin
3.3333000000000003e-11
>>> fmax
3333300000000000.0
```

answer değişkeni

→ answer (_) değişkeni

```
>>> 5 + 6
11
>>> _ + 22
33
```

gerçel sayılar

→ gerçel sayılar

```
>>> 10./3 + 20./3
10.0
```

dizgi mi? sayı mı?

→ dizgi x sayı

```
>>> print 12 + 34
46
>>> print "12" + "34"
1234
```

ileri düzey

→ değişken ismini dizgiden almak

```
1  >>> kim = "ahmet"
2  >>> vars()[kim] = 35
3  >>> print ahmet
4  35
5  >>> ahmet
6  35
```

ileri düzey

→ değişken ismini dizgiden almak

```
1  >>> kim = "ahmet"
2  >>> vars()[kim] = 35
3  >>> print ahmet
4  35
5  >>> ahmet
6  35
```

→ veya böyle

```
1  >>> kim = raw_input("Kimin yasi?")
2  Kimin yasi ? ahmet
3  >>> vars()[kim] = 43
4  >>> ahmet
5  43
```

2.4 Cümleler

cümle: Python yorumlayıcısınca işlenebilecek yönerge

→ şimdiye kadar print ve atama cümlesini gördük

→ cümleyi yaz (kabuğa), Python işlesin (yorumlasın), sonucu versin (eğer varsa)

→ print cümlesinin sonucu bir değerdir

→ atama cümlesi herhangi bir sonuç üretmez

```
>>> print a = 3
  File "<stdin>", line 1
    print a = 3
          ^
SyntaxError: invalid syntax
```

cümle: betik

→ betikler ardı ardına gelen cümlelerden oluşur

→ her bir cümle işletilir (sırayla) (sırayı bozan özel yapılar var, sonra)

→ her bir cümlenin sonuçları gösterilir

→ örnek

```
1  print 1
2  x = 2
3  print x
```

→ şu çıktı üretilir sırayla

```
1  1
2  2
```

→ atama cümlesi çıktı üretmez!

Deyimler

deyim: değerlerden (ör. 1), değişkenlerden (ör. yas) ve işleçlerden (ör. +) oluşan yapı.

→ eğer bir deyimi kabuğa yazarsanız ve sonlandırma (yürüt) tuşuna (<enter>) basarsanız

→ o deyim **değerlendirilir** (hesaplanır) ve sonucu gösterilir

```
>>> 1 + 4
5
```

→ değer de başlı başına bir deyimdir

→ değişken de öyle

```
>>> 17
17
>>> a
20
```

deyim: atama, değerlendirme

→ değişkenlere değer atanır (satır 1)

→ değerlendirilir (satır 2, 4)

```
1  >>> mesaj = "merhaba, dunya"
2  >>> mesaj
3  'merhaba, dunya'
4  >>> print mesaj
5  merhaba, dunya
```

betik

→ betik dosyasında değerler tek başına çıktı üretmez!

```
17
3.2
"merhaba, dunya"
1 + 1
```

→ peki ekranda görebilmek için ne yapmalıyız?

İşleçler ve işlenenler

→ işlem = işlenenler + işleç

işleç: işlemi belirten, toplama, çarpma vb. hesaplamaları temsil eden özel _sembollerdir_

işlenen: işleme giren değerler.

→ örnek

```
20 + 32
hour - 1
hour*60 + minute
(5 + 9) ** (15 - 7)
```

tamsayı bölme

→ eğer her iki işlenen tamsayı ise tamsayı bölme adlanır

```
>>> dakika = 59
>>> dakika / 60
0
```

→ işlenenler neyse (tamsayı) sonuç aynı (tamsayı)

→ sonucun gerçel olması istenirse işleneni gerçele çevir (4. bölüm)

işleçlerin sırası

- → deyim içerisinde birden fazla sayıdaki işlecin hangi sırayla değerlendirileceği
- → öncelik sırası (kuralı) ile belirlenir
- → P arantez en yüksek öncelik
- → Ü s alma. Aşağıdaki sonuçları söyleyin (3? 27?)

```
>>> 3 * 1 ** 3
```

- → Ç arpma ve B ölme aynı öncelikte, T oplama ve Ç ıkarmadan daha yüksek öncelikli
- → aynı önceliklilerde **soldan sağa** kuralı

karakter dizisi üzerindeki işlemler

→ genel olarak dizgiler üzerinde matematiksel işlem geçersizdir

```
>>> mesaj = "hello" # mesaj = "12"
>>> mesaj - 1
>>> mesaj / 123
>>> mesaj * "bar"
>>> mesaj + 2
```

dizgi: birleştirme

→ + işleci dizgilerde birleştirme (concatenation),ardı ardına ekleme

```
ad = "Ahmet"
soyad = "Kilic"
print "Merhaba" + ad + soyad
```

→ sonuç: Merhaba Ahmet Kilic olacaktır

dizgi: çoğaltma

→ * işleci de dizgilerle çalışır. Tekrarlama veya çoğaltma işlemi

```
>>> "Python'u " + "cok" * 5 + " seviyoruz"
"Python'u cokcokcokcokcok seviyoruz"
```

dizgi - sayı benzerlik

→ 4 * 3, deyimini 4 + 4 + 4 olarak düşünebiliriz

→ "cok" * 3, deyimi de benzer olarak "cok" + "cok" + "cok" olur

girdi

→ kullanıcıdan (klavyeden) girdi almak için: input, raw_input

```
ad = raw_input("Lutfen adinizi giriniz: ")
no = input("Lutfen ogrenci numaranizi giriniz: ")
print "%s nin numarasi %s dir"%(ad, no)
```

→ çıktısı

```
$ python 02_girdi.py
Lutfen adinizi giriniz: nurettin
Lutfen ogrenci numaranizi giriniz: 987
nurettin nin numarasi 987 dir
```

→ input girdilerinde sayısal ifade de girebilirsiniz

girdi: type

→ aşağıdaki her bir durum için çıktılar neler olabilir

```
>>> x = input()
3.14
>>> type(x)
```

?	?	?

```
>>> x = raw_input()
3.14
>>>type(x)
```

?	?	?

yorumlar

→ doğal dillerdeki notlara ve açıklamalara benzer olarak kodun belirli bölgesini

→ açıklamada kullanılan yapılara **yorum** denilir

→ tek satırlık yorumlar # ile başlar

```
# gecen zamanin yuzdesi
yuzde = (dakika * 100) / 60

yuzde = (dakika * 100) / 60  # dikkat: tamsayi bolme
```