Python Programming

Giriş

PYTHON DİLİNİN ÖZELLİKLERİ

- * Nesneye yönelikdir.
- * Özgürdür.
- * Yorumlamalı ve derlemelidir. Her ne kadar sadece yorumlamalı gözükse de "byte code" biçiminde
- derleme yapılır.
- * Tasınabilir.
- * Güçlüdür. Đhtiyacınız olabilecek hemen hemen tüm özellikler mevcuttur.
- * Hızlıdır.
- * Ticari uygulamalar gelistirmeye uygundur. Bunun için kodu bytecode seklinde verebilirsiniz ya da
- "py2c" ile C modülüne çevirip derleyebilirsiniz.
- * Yazılımı kolaydır.
- * Öğrenmesi kolaydır.

Değişkenler

- Değişkenler, bir değeri temsil eden bir sembol veya isimdir.
- Değişken, değişebilen bir değerdir.
  - Değişkenler sayısal değerleri, karakterleri, karakter dizilerini veya bellek adreslerini temsil edebilir.

Bağımlı - BağımsızDeğişkenler

- Farklılık gösteren özelliklere değişken denir.
- Bir özellik her gözlemde aynı değeri alıyorsa, yani gözlemden gözleme değişmiyorsa bu duruma sabit denir. Örneğin, pi değeri değişmeyen bir değer olacağı için sabit olarak tanımlanır. Y=at+b. a ve b sabit katsayılardır, t ve y ise değişkendir. y, t'ye bağlı bir değişkendir. Burada t, bağımsız bir değişkendir.
- **Özellik: Değişkenin bir yönünü ya da niteliğini bildiren kavramdır.**

Nedensillerine göre bağımlı ya da bağımsız değişkenler.

- **Bağımlı ve bağımsız değişkenler:** Bazı değişkenler başka bir değişkene bağlı olmadan artar ya da azalırlar, yani değişirler. Bu tür değişkenlere bağımsız değişken adı verilir. Bazı değişkenler ise başka bir değişkene bağlı olarak değerler alırlar ve bağımlı değişken olarak adlandırılırlar.

Sıklık (Frekans) Dağılımı: Verilerin gösterdiği dağılıma sıklık (frekans) dağılımı denir. Aynı peryoda sahip sinyalin bütün içerisinde kaç kez tekrar etmesidir.

Sürekli – Süreksiz Değişkenler

Aldıkları değerlere göre sürekli ya da süreksiz değişkenler.

- **Kesikli (Süreksiz) Değişken:** Tanımlı olduğu aralıklarda ayrık değerler alan değişkenlerdir. Dizilerdir. [1,3,5,2,9,5,4, ...]
- **Sürekli Değişken:** Tanımlı olduğu aralıkta tüm değerleri (sonsuz sayıda) alabilen değişkenlerdir. Y(t)=Asin(wt+fi),
- Burada A: genlik, fi: faz açısı, w=2*pi*f, f: frekans [1/sec: Hz]

Nicel – Nitel Değişkenler

Özelliklerine göe nicel ve nitel değişkenler.

- **Nicel (Kantitatif) Değişkenler:** Ölçüm sonucu değerleri sayısal özellikler belirten değişkenlerdir. Sayılabilir veya ölçülebilir büyüklüklerdir.
- **Nitel (Kalitatif) Değişkenler:** Karakteristik özellikleri, durumları ve pozisyonları belirten değişkenlerdir. Sayılamayan, birimlendirilemeyen ve ölçülebilir olmayan büyüklüklerdir. Bir şeyin nasıl olduğunu belirten, onu başka şeylerden ayıran özelliktir (sıfat).
- Qualitative: Words, images, observations, conversations, photographs
- Quantitative: Quantitative – Numbers, tests, counting, measuring; Discrete, Continuous

Fonksiyon değerleri

- "f(t)" gibi bir ifade, t noktasında değerlendirilen f fonksiyonunun değeri anlamına gelir.
- Mühendisler f fonksiyonuna atıfta bulunmak için sıklıkla "f(t)" gibi bir ifade kullanırlar (t noktasında değerlendirilen f'nin değeri yerine) ve bu özensiz gösterim bazı durumlarda sorunlara (örn. Belirsizlik) yol açabilir. Bu nedenle bir fonksiyon ve değeri arasında net bir ayrım yapmaya dikkat edilmelidir.

Örnek (gösterimin anlamı):

- f ve g bir gerçek değişkenin gerçek değerli fonksiyonlarını; t, ile rasgele bir gerçek sayıyı gösterelim.
- H'nin bir sistem operatörünü (bir fonksiyonu bir fonksiyona eşleyen) göstersin.
- f ve g fonksiyonlarının toplanmasıyla oluşturulan f + g miktarı da bir fonksiyondur.
- f(t) + g(t) miktarı bir sayıdır, yani şunların toplamı: t'de değerlendirilen f fonksiyonun değeri ile t'de değerlendirilen g fonksiyonunun değerinin toplamıdır.
- Hx miktarı bir fonksiyondur, yani sistem girdisi x fonksiyonu olduğunda H ile temsil edilen sistem tarafından üretilen çıktıdır.
- Hx(t) miktarı bir sayıdır, yani Hx fonksiyonunun t'de değerlendirilen değeridir.

Neden Python

- Değişkenler
- İşlemler: Aritmetik işlemler, mantıksal işlemler, Karşılaştırma
- Döngüsel komutlar: if, for, while, ...
- Yazılım modülleri: Kütüphane
- Otonom – yapay zeka; Yazılım modüllerinden akıllı algoritmalar elde edilir.
- Sistemlerin performansı çalışandan bağımsız olarak iyileştirme yapabilecektir.

Python'un Büyüsü

- Bu komutlara deyimler denir.
- `x = 5`
- `print(x)`
- `print("Merhaba, Dünya")`
- `print(2+3)`

Python'un Büyüsü

- Python isteminden çıktığımızda, tanımladığımız işlevler sona eriyor!
- Programlar genellikle tekrar tekrar kullanılabilmeleri için diske kaydedilen işlevlerden, modüllerden veya hertürlü yazılımlardan oluşur.
- Modül dosyası, metin düzenleme yazılımında oluşturulan ("düz metin" olarak kaydedilen) işlev tanımlarını içeren bir metin dosyasıdır.
- Bir programlama ortamı, programcıların program yazmasına yardımcı olmak için tasarlanmıştır ve genellikle otomatik girintileme, vurgulama vb. içerir.

Python'un Büyüsü

- Genellikle ortak bir sorunu çözen birkaç ifade birlikte yürütülmek istenen program yazılmak istendiğinde fonksiyon kullanılır.

- ```
  def merhaba():
      print("Merhaba")
      print("Bilgisayarlarin da sevilmeye ihtiyaclari var")
  ```

- İlk satır Python'a merhaba adında yeni bir fonksiyon tanımladığımızı söyler.
- Aşağıdaki satırlar, merhaba isimli fonksiyonun parçası olduklarını göstermek için sağa doğru içeriye girintilidir.
- Fonksiyon tanım komutdan sonra gelecek olan, aşağıdaki satırlar, fonksiyonun parçası olduklarını göstermek için sola doğru içeriye girintilidir.
- Boş satır (iki kez enter tuşuna basın), Python'un yazılımın bittiğini bilmesini sağlar.
  ```

Python'un Büyüsü

```
def hello():
    x = 5
    print(x)
    print("Merhaba, Dünya")
    print(2+3)
    print("Bilgisayarlarin da sevilmeye
ihtiyaclari var")

hello()
```

- Fonksiyon tanımlandıktan sonra Python'a fonksiyonu gerçekleştirmesini söylemek gerekir!
- Sadece fonksiyonun adı yazılarak program çalıştırılır.

Python'un fonksiyonu

- Komutlar, fonksiyonu tanımlayan ismin parantezi arasına yerleştirilen parametre adı verilen değiştirilebilir ifadlere sahip olduklarında, fonksiyonun çıktısı özelleştirilir.
- ```
  def greet(person):
          print("Hello",person)
          print ("How are you?")
  ```

```
def greet(person):
    print("Hello",person)
    print ("How are you?")

greet("Mary")
```
```

Python'un Büyüsü

```
# File: chaos.py
# A simple program illustrating chaotic behavior

def main():
    print("This program illustrates a chaotic function")
    x = eval(input("Enter a number between 0 and 1: "))
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print(x)

main()
```

- Kod yazılımını kaydettiğimizde bunun bir Python programı olduğunu belirtmek için filename.py kullanacağız.
- Yazılımda «main» adında bir fonksiyon tanımlandığında, «main()», Python'a kodu çalıştırmasını söyler.

Python'un Büyüsü

```
>>>
This program illustrates a chaotic function
Enter a number between 0 and 1: .5
0.975
0.0950625
0.335499922266
0.869464925259
0.442633109113
0.962165255337
0.141972779362
0.4750843862
0.972578927537
0.104009713267
>>>
```

Bir Python Programının İçinde

```
# File: chaos.py
# A simple program illustrating chaotic behavior
```

- # ile başlayan satırlara yorum denir
- İnsan denilen iki ayaklı okuyucularına yöneliktir ve Python tarafından dikkate alınmaz
- Python metni #'den satırın sonuna kadar atlar

Bir Python Programının İçinde

```
def main():
```

- Main adlı bir fonksiyonun tanımının başlangıcı
- Programımız sadece bu modüle sahip olduğundan, main fonksiyonu olmadan da yazılabilirdi.
- Bununla birlikte, main kullanımı alışılmış bir durumdur.

Bir Python Programının İçinde

```
print("This program illustrates a chaotic function")
```

- Bu satır, Python'un programı tanıtan bir mesaj yazdırmasına neden olur.

Bir Python Programının İçinde

```
x = eval(input("Enter a number between 0 and 1: "))
```

- x bir sayısal değişken tanımlar.
- Bir değişken, bir değere daha sonra başvurabilmemiz için bir ad atamak için kullanılır.
- Alıntılanan bilgi görüntülenir ve yanıtta yazılan sayı x'te saklanır.

Bir Python Programının İçinde

```
for i in range(10):
```

- For bir döngü yapısıdır
- Bir döngü, Python'a aynı şeyi defalarca tekrarlamasını söyler.
- Bu örnekte aşağıdaki kod 10 kez tekrarlanacaktır.
- Döngüdeki i 0 ile 9 arasında değişir.
- i=0,1,2,3,4,5,6,7,8,9

Bir Python Programının İçinde

```
for i in range(10):
    x = 3.9 * x * (1 - x)
    print(x)
```

- For ile başlayan ifadeden sonraki satırlar döngünün gövdesidir.
- Döngünün gövdesi, döngü boyunca her seferinde tekrarlanan şeydir.
- Döngünün gövdesi sağa girinti yoluyla tanımlanır.
- Döngünün etkisi, sağa girintili satırları 10 kez tekrarlayacaktır.

Bir Python Programının İçinde

```
for i in range(10):
    x = 3.9 * x * (1 - x)
    print(x)
```

- Bunlar eşdeğerdir!

```
x = 3.9 * x * (1 - x)
print(x)
x = 3.9 * x * (1 - x)
print(x)
x = 3.9 * x * (1 - x)
print(x)
 x = 3.9 * x * (1 - x)
print(x)
 x = 3.9 * x * (1 - x)
 print(x)
 x = 3.9 * x * (1 - x)
 print(x)
 x = 3.9 * x * (1 - x)
 print(x)
 x = 3.9 * x * (1 - x)
 print(x)
 x = 3.9 * x * (1 - x)
 print(x)
 x = 3.9 * x * (1 - x)
 print(x)
```

Bir Python Programının İçinde

```
x = 3.9 * x * (1 - x)
```

- Buna atama bildirimi denir
- “=” işaretinin sağ tarafındaki kısım (RHS:right-hand side; LHS: Left-hand side) matematiksel bir ifadedir.
- * çarpmayı belirtmek için kullanılır
- RHS'deki değer hesaplandıktan sonra, x'e geri depolanır (atanır)

Bir Python Programının İçinde

```
main()
```

- Python'a kodu main fonksiyonunda çalıştırmasını söyler.

Kaos ve Bilgisayarlar

- The chaos.py program:

```
def main():
    print("This program illustrates a chaotic function")
    x = eval(input("Enter a number between 0 and 1: "))
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print(x)
main()
```

- Herhangi bir giriş için, 0 ile 1 arasında rastgele görünen 10 sayı döndürür
- Görünüşe göre x'in değeri kaotik

Kaos ve Bilgisayarlar

- Program tarafından hesaplanan fonksiyon, k'nin 3.9 olduğu

$$k(x)(1—x)$$

genel formuna sahiptir.

- Bu tür bir işlev, lojistik bir işlev olarak bilinir.
- Belirli türdeki kararsız elektronik devreleri modeller.
- Başlangıç değerindeki çok küçük farklılıklar, çıktıda büyük farklılıklara neden olabilir.

Kaos ve Bilgisayarlar

- Input: 0.25

0.73125
0.76644140625
0.698135010439
0.82189581879
0.570894019197
0.955398748364
0.166186721954
0.540417912062
0.9686289303
0.118509010176

- Input: 0.26

0.75036
0.73054749456
0.767706625733
0.6954993339
0.825942040734
0.560670965721
0.960644232282
0.147446875935
0.490254549376
0.974629602149

Veri tipi dönüştürmeli giriş

```
x = int(input("Input an integer: "))
y = float(input("Input a float: "))
print (x, y)
```

- Şimdi bir dairenin alanını hesaplamak için programımızı değiştirebiliriz, böylece çapı girebiliriz:
- Kütüphane, math.pi: matematik kütüphanesinde kayıtlı ya da hesap edilmiş pi değerini kullanılmasına izin verilir.

```
# Calculate area of a circle
import math
d = float(input("Diameter: "))
A = math.pi *(d/2)**2
print ("Area = ", A)
```

Diameter: 25
Area = 490.873852123

Not: #: Programın başındaki metin ne yaptığının bir açıklamasıdır. Her programın ne yaptığına dair kısa bir açıklaması olmalıdır.

Örnek Program: Sıcaklık Dönüştürücü

```
#convert.py
# A program to convert Celsius temps to Fahrenheit
# by: Susan Computewell

def main():
    celsius = eval(input("What is the Celsius temperature? "))
    fahrenheit = (9/5) * celsius + 32
    print("The temperature is ", fahrenheit, " degrees Fahrenheit.")
main()
```

Örnek Program

```
# ismi selamlama
isim = input('İsminiz nedir : ')
print("Merhaba "+isim)

a1= input('ilk rakami girin : ')
b1= input('ikinci rakami girin : ')
c=a1+b1
print("Sonuc: ", c)

toplam=float(a1)+float(b1)
print("Toplam :{0} ".format(toplam))
```

Örnek: Sınav Geçme Notu Belirleme

```
vize = input('Vize Notunuz : ')
final = input('Final Notunuz : ')
devam = input('Devam gun sayisi : ')
ortalama=(float(vize)*0.30)+(float(final)*0.475)+(float(devam)*1.5)
print("Ortalama :{0} ".format(ortalama))
if(int(ortalama)>=60):
    print("Geçtiniz")
else:
    print("Kaldiniz")
```

Örnek: Sayının tek mi çift mi; pozitif mi negatif mi olduğunu belirleme

```
sayi = input('Sayi : ')
if(int(sayi)%2==0):
    print("Sayi Çift")
else:
    print("Sayi Tek")

if(int(sayi)<0):
    print("Sayi Negatif")
elif(int(sayi)>0):
    print("Sayi Pozitif")
else:
    print("Sayi Sifir")
```

Yazılım Geliştirme Süreci

Yazılım Geliştirme Süreci

- Problem Analiz Edilir
- Spesifikasyonlar Belirlenir
- Tasarım Yaratılır
- Tasarım Uygulanır
- Program Test Edilir veHata Ayıklanır
- Programın Bakımı Yapılır

Program Unsurları

- **İsimler, katsayılar**
  - Her tanımlayıcı ingilizce bir harf veya alt çizgi (“_”) ile başlamalı ve ardından herhangi bir harf, rakam veya alt çizgi dizisi gelmelidir.
  - Tanımlayıcılar büyük/küçük harfe duyarlıdır.ayarlara (celsius, fahrenheit), modüllere (main, convert) vb. adlar verilir.
  - Bu adlara koruyucu denir
  - Her gelen bir harf veya alt çizgi (“_”) ile başlamalı ve ardından herhangi bir harf, rakam veya alt çizgi dizi gelmelidir.

Program Unsurları

– Bunların hepsi farklı, geçerli isimler
  - X
  - Celsius
  - Spam
  - spam
  - spAm
  - Spam_and_Eggs
  - Spam_And_Eggs

Program Unsurları

- Bazı tanımlayıcılar Python'un kendisinin bir parçasıdır. Bu tanımlayıcılar ayrılmış kelimeler olarak bilinir. Bu, programınızda bir değişken vb. adı olarak kullanamayacağınız anlamına gelir.
- and, del, for, is, raise, assert, elif, in, print, etc.

Program Unsurları

- İfade
  - Yeni veri değerleri üreten veya hesaplayan kod parçalarına ifadeler denir.
  - Değişmez değerler, belirli bir değeri temsil etmek için kullanılır; 3.9, 1, 1.0
  - Basit tanımlayıcılar da ifadeler olabilir.

Program Unsurları

```
>>> x = 5
>>> x
5
>>> print(x)
5
>>> print(spam)

Traceback (most recent call last):
  File "<pyshell#15>", line 1, in -toplevel-
    print spam
NameError: name 'spam' is not defined
>>>
```

- NameError, kendisine bir değer atanmamış bir değişkeni kullanmaya çalıştığınızda oluşan hatadır.

Program Unsurları

- Daha basit ifadeler operatörler kullanılarak birleştirilebilir.
- +, -, *, /, **
- Boşluklar bir ifade içinde ilgisizdir.
- Normal matematiksel öncelik geçerlidir.
- ((x1 – x2) / 2*n) + (spam / k**3)

Operators

- Arithmetic
- Comparison
- Assignment
- Logical
- Bitwise
- Membership
- Identity

+	-	*	//	/	%	**

==	!=	>	<	>=	<=

=	+=	-=	*=	//=	/=	%=	**=

and	or	not

&	|	^	~	>>	<<

in	not in

is	is not

Supported operators

Supported operators:

Operator		Example	Explication
+, - *, /	add, substract, multiply, divide		
%	modulo	25 % 5 = 0 84 % 5 = 4	25/5 = 5, remainder = 0 84/5 = 16, remainder = 4
**	exponent	2**10 = 1024	
//	floor division	84//5 = 16	84/5 = 16, remainder = 4

*, /, %, //	Çarpma, bölme, mod alma, alta yuvarlayarak bölme.
+, -	Toplama ve çıkarma
<=, <, >, >=	Kıyaslama operatörleri, küçük-eşit, küçük, büyük, büyük-eşit
<>, ==, !=	Eşitlik operatörleri,
=, %=, /=, //=, -=, +=, *=, **=	Atama operatörleri
is, is not	Kimlik operatörleri
in, not in	Üyelik operatörleri
not, or, and	Mantıksal operatörler

Logic

Operator	Meaning	Example	Result
==	equals	1 + 1 == 2	True
!=	does not equal	3.2 != 2.5	True
<	less than	10 < 5	False
>	greater than	10 > 5	True
<=	less than or equal to	126 <= 100	False
>=	greater than or equal to	5.0 >= 5.0	True

Operator	Example	Result
and	9 != 6 and 2 < 3	True
or	2 == 3 or -1 < 5	True
not	not 7 > 0	False

- Mantıksal ifadeler, mantıksal işleçlerle birleştirilebilir.

Object types

Object type	Type class name	Description	Example
Integer	int	Signed integer, 32 bit	a = 5
Float	float	Double precision floating point number, 64 bit	b = 3.14
Complex	complex	Complex number	c = 3 + 5j c= complex(3,5)
Character	chr	Single byte character	d = chr(65) d = 'A' d = "A"
String	str	List of characters, text string	e = 'LTAM' e = "LTAM"

Değişkenler

```
x = 4 # x tam sayı tipinde
x = "Fikri" # x artık string tipinde
print(x)
```

- Sayılar için + karakteri matematiksel bir operatör olarak çalışır:

```
x = 5
y = 10
print(x + y)
```

- Eğer bir karakter dizisini(str) ve sayıyı(int) + operatörü ile birleştirmeye çalışırsanız; Python karakter dizisi ve sayıyı birleştiremeyeceğine göre hata verecektir.

```
x = 5
y = "John"
print(x + y)
```

- Başka bir değişkene başka bir değişken eklemek için + operatörünü kullanabilirsiniz.

```
x = "awesome"
y = "Peter is " + x
print(y)
```

Program Unsurları

- Çıktı Tabloları
  - Bir print deyimi herhangi bir sayıda ifadeyi yazdırabilir.
  - Ardışık yazdırma ifadeleri ayrı satırlarda görüntülenecektir.
  - Çıplak bir baskı, boş bir satır yazdırır.

Program Unsurları

```
print(3+4)
print(3, 4, 3+4)
print()
print(3, 4, end=" "),
print(3 + 4)
print("The answer is", 3+4)
```

```
7
3 4 7


3 4 7
The answer is 7
```

Atama İfadeleri

- Basit Atama
- <değişken> = <ifade> değişken bir tanımlayıcıdır, ifade bir ifadedir
- RHS'deki ifade, daha sonra LHS'de adlandırılan değişkenle ilişkilendirilen bir değer üretmek için değerlendirilir.

Atama İfadeleri

- x = 3.9 * x * (1-x)
- fahrenheit = 9/5 * celsius + 32
- x = 5

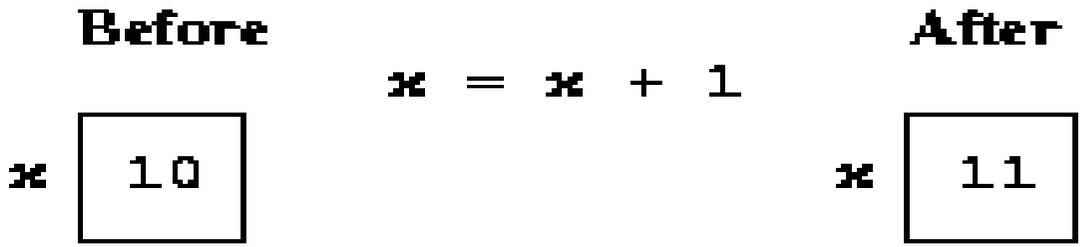
Atama İfadeleri

- Değişkenler istediğiniz kadar yeniden atanabilir!

```
>>> myVar = 0
>>> myVar
0
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
>>>
```

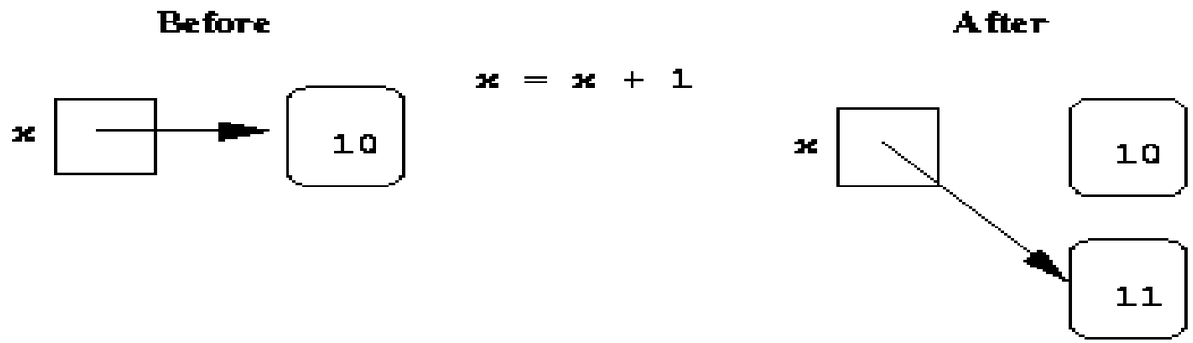
Atama İfadeleri

- Değişkenler, içine değer koyabileceğimiz bir kutu gibidir.Bir değişken değiştiğinde eski değer silinir ve yerine yenisi yazılır.



Atama İfadeleri

- Teknik olarak, bu atama modeli Python için basittir.
- Python bu bellek konumlarının (kutularının) üzerine yazmaz.
- Bir değişken atamak, daha çok bir değerin üzerine "yapışkan not" koyup "bu x" demek gibidir.



Eval: Giriş Atama

- Eval fonksiyonun amacı, kullanıcıdan değeri girdi almak ve bunu bir değişkende saklamaktır.
- Eval işlevini, metin dizesi veya sayısal ifade bir değerle sonuçlandı olarak değerlendirmek için kullanabilirsiniz
- Evaluate kelimesinin kısaltmasından gelen eval() fonksiyonu, basit anlatımla kendisine verilen karakter dizisini değerlendirir ve bir sonuç verir.
- <variable> = eval(input(<prompt>))

Örnek: Giriş Atama

```
#kullanıcıya matematiksel işlem yapabileceği bir program yazdığımızı varsayalım
#bunun için, girdi değişkenine kullanıcıdan karakter kümesi olarak alacağımız işlemi atayalım
girdi=input("Lütfen sonucunu bulmak istediğiniz matematiksel işlemi yazın. + - * / matematiksel işaretlerini kullanabilirsiniz.")
#girdi değişkenini eval() fonksiyonuyla işleyelim ve işlem sonucunu sonuc değişkenine atayalım
sonuc=eval(girdi)
#Sonucu yazdıralım
Print("Verdiğiniz matematiksel işlemin sonucu:", sonuc)
```

- Girdi olarak 87*48 yazılırsa, eval() fonksiyonu bu karakter kümesini değerlendirip, bunu bir çarpma işlemi olarak algılar ve doğrudan sonucu yazar. eval() fonksiyonu kullanıcından alınan herhangi bir komutu çalıştırmak üzerine kurulmuş. Eğer herhangi bir kısıt koymazsanız, programınıza erişen kişi bilgisayarınızda dosya yaratabilir, dosyaların içeriğini değiştirebilir, dosyaları silebilir ve birçok işlemi rahatlıkla yapabilir.

Giriş Atama

- Önce bilgi istemi yazdırılır.
- Giriş kısmı, kullanıcının bir değer girip <enter> tuşuna basmasını bekler.
- Girilen ifade, onu bir karakter dizisinden Python değerine (bir sayı) dönüştürmek için değerlendirilir.
- Değer değişkene atanır.

Eşzamanlı Çoklu Atama

- Aynı anda birkaç değer hesaplanabilir
- <var>, <var>, … = <expr>, <expr>, …
- Sağ taraftaki (RHS'deki) ifadeler değerlendirilir ve bunlar Sol taraftaki (LHS'deki) değişkenlere atanır.

Eşzamanlı Çoklu Atama

- sum, diff = x+y, x-y
- Bunu x ve y'nin değerlerini değiştirmek için nasıl kullanabilirsiniz?
  - Bu neden çalışmıyor?
    x = y
    y = x
- Burada x’in ilk değeri kaybolur. Geçici bir değişken kullanabiliriz…

Eşzamanlı Atama

- Python'da iki değişkenin değerlerini kolayca değiştirebiliriz!
  - x, y = y, x

```
>>> x = 3
>>> y = 4
>>> print (x, y)
3 4
>>> x, y = y, x
>>> print (x, y)
4 3
```

Eşzamanlı Atama

- Aynı fikri, tek bir girdi ifadesinden birden fazla değişken girmek için kullanabiliriz!
- Girişleri ayırmak için virgül kullanılır.

```
def siparis():
    ekmek, su = eval(input("ekmek ve su adetlerini giriniz: "))
    print("siz", ekmek, "adet ekmek ve", su, "adet su siparis ettiniz")

siparis()
ekmek ve su adetlerini giriniz: 6,9
siz 6 adet ekmek ve 9 adet su siparis ettiniz
```

Aritmetik İşlemler

- +,-,*,/ islemleri mevcuttur. Yalnız bunları kullanırken tip uyusmazlığına dikkat etmek gerekir. '**' üs almak için kullanılır.
- A=8**2
- print(A)

64

- B=(3 + 3j) + (4 -1j)
- print(B)

(7+2j)

- a=9;b="5"
- print(a+int(b))

14

- print(str(a)+b)
- print(len(str(a)+b))

95

2

Kesin Döngüler

- Belirli bir döngü belirli sayıda yürütür, yani Python döngüyü başlattığında tam olarak kaç yineleme yapacağını bilir.
- for <var> in <sequence>:
  <body>
- Gövdenin başı ve sonu girinti ile gösterilir.
- for'dan sonraki değişkene döngü indeksi denir. Ardışık her değeri sırayla alır.

```
for i in [7,1,82,3]:
    print (i)
```

```
7
1
82
3
```

```
for odd in [1, 3, 5, 7]:
    print(odd*odd)
```

```
1
9
25
49
```

Kesin Döngüler

- for i in [7,1,82,3]:
- print (i)
- print(list(range(10)))
- [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- `range, 0'dan başlayarak bir sayı dizisi oluşturan yerleşik bir Python kodunun işlevidir.`
- `list, diziyi açık bir listeye dönüştüren yerleşik bir Python işlevidir.`
- `Döngünün gövdesi 10 kez yürütülür.`

Örnek Program: Gelecek Değer

- Analiz
  - Banka hesabına yatırılan para faiz getirir.
  - Bundan 10 yıl sonra hesabın değeri ne kadar olacak?
  - Girdiler: anapara, faiz oranı
  - Çıktı: 10 yıldaki yatırımın değeri

Örnek Program: Gelecek Değer

- Tanımlamalar
  - Kullanıcı yatırım yapmak için ilk tutarı girer, anapara
  - Kullanıcı yıllık faizi girer, **faiz**
  - Spesifikasyonlar şu şekilde temsil edilebilir…

- Program: Gelecek Değeri
- Girdiler
  - **Anapara**, Anapara yatırılan para miktarı, dolar cinsinden
  - **Faiz** , Ondalık sayı olarak ifade edilen yıllık faiz.
- Çıktı 10 yıl sonraki yatırımın değeri
- Bir yıl sonraki İlişki, Anapara * (1 + **Faiz/100**) ile verilmiştir. Bunun 10 kez yapılması gerekiyor.

Örnek Program: Gelecek Değer

Tasarım

- Bir tanıtım yazdırın
- Anapara tutarını girin (Anapara)
- Yıllık Faiz girin (Faiz)
- 10 kez tekrarlayın:anapara = Anapara * (1 + Faiz/100)
- Anaparanın değerini çıkar

Örnek Program: Gelecek Değer

Uygulama

- Her satır bir Python satırına çevrilir (bu durumda)
- Bir tanıtım yazdırın
  - Print("Bu program gelecegi hesaplar")
  - Print("10 yillik bir yatirimin degeri.")
- Anapara tutarini girin
  - Anapara= eval(input("İlk anaparayi girin: "))
- Yıllık faizi girin
  - Faiz = eval(input("Yillik faizi giriniz: "))
- 10 kez tekrarlayın:
  - for i in range(10):
- Anapara = Anapara * (1 + Faiz/100) ifadesini hesaplayın.
  - Anapara = Anapara * (1 + Faiz/100)
- Anaparanın 10 yıl sonundaki değerini yazdırın
  - Print("10 yildaki deger:", Anapara)

Örnek Program: Gelecek Değer

```
# Anapara.py
# Bir yatırımın değerini hesaplamak için bir program# 10 yılı geleceğe taşıdı

def main():
    print("Bu program, 10 yillik bir yatirimin gelecekteki değerini hesaplar.")

    Anapara = eval(input("Anaparayı Girin: "))
    Faiz = eval(input("Yillik faizi girin: "))

    for i in range(10):
        Anapara = Anapara * (1 + Faiz/100)

    print ("10 yildaki değeri:", Anapara)

main()
```

Örnek Program: Gelecek Değer

```
>>> main()
Bu program, 10 yillik bir yatirimin gelecekteki değerini hesaplar.
Anaparayı Girin: 100
Yillik faizi girin: : .3
The value in 10 years is: 13,439.16
>>> main()
Bu program, 10 yillik bir yatirimin gelecekteki değerini hesaplar.
Anaparayı Girin: 100
Yillik faizi girin: : 10
The value in 10 years is: 259.3742
```

Sayılarla Hesaplama

Amaç

- Veri türleri kavramını anlamak.
- Python'daki temel sayısal veri tiplerine aşina olmak.
- Sayıların bir bilgisayarda nasıl temsil edildiğine ilişkin temel ilkeleri anlamak.
- Python matematik kütüphanesini kullanabilmek.
- Akümülatör program modelini anlamak.
- Sayısal verileri işleyen programları okuyabilir ve yazabilir.

Python Data Types

Python has five standard Data Types:

- Numbers
- String
- List
- Tuple
- Dictionary

Different types of data

- Numbers
  - Whole number, fractional number, …
- Text
  - ASCII code, unicode
- Audio
- Image and graphics
- Video

How can they all be represented as binary strings?

Numbers

- Integer
- Real, Ondalık sayılar
- Exponential
- Kompleks
- Karakterler

```
-3, 3
0.0001, -0.25
1.60210e-20
5 + 2i
'myimage'
```

Types of Data - Sayısal Veri Türleri

- Sayısal veri türleri arasında tamsayılar ve değişkenler bulunur. Bir kayan noktalı (kayan nokta olarak bilinir) sayı, ondalık nokta değeri 0 olsa bile ondalık basamaklara sahiptir. Örneğin: 1,13, 2,0 1234,345.
- Hem tamsayıları hem de kayan noktalı sayıları içeren bir sütunumuz varsa, Pandas sütunun tamamını kayan veri türüne atar, böylece ondalık basamaklar kaybolmaz.Bir tamsayının hiçbir zaman ondalık noktası olmaz.
- Böylece 1.13, 1 olarak saklanır. 1234.345, 1234 olarak saklanır.
- Python'da 64 bit tamsayı anlamına gelen Int64 veri tipini sıklıkla göreceksiniz. 64, basitçe, her bir "hücrede" kaç basamak saklayabileceği ile etkili bir şekilde ilgili olan, her bir hücrede veri depolamak için tahsis edilen hafızayı ifade eder. Alanın önceden tahsis edilmesi, bilgisayarların depolama ve işleme verimliliğini optimize etmesine olanak tanır.

Types of Data -Karakter Veri Türleri

- Pandalarda Object olarak bilinen stringler, sayıları ve/veya karakterleri içeren değerlerdir.
- Örneğin, bir dize bir kelime, bir cümle veya birkaç cümle olabilir.
- Bir Pandas nesnesi, "arsa1" gibi bir olay örgüsü adı da olabilir. Bir dize ayrıca sayı içerebilir veya sayılardan oluşabilir.
- Örneğin, '1234' bir dizi olarak saklanabilir. '10.23' gibi. Ancak sayı içeren diziler matematiksel işlemlerde kullanılamaz!

Katsayılar, Değişkenler
Bağımlı ve Bağımsız Değişkenler

- F(t)=at^2+bt+c ifadesinde
- Katsayılar: a, b, c
- Değişkenler: t, f(t)
- Bağımsız değişken: t
- Bağımlı değişken: f(t)

Sayısal Veri Türleri

- Bilgisayar belleklerinde saklanan ve işlenen bilgilere veri denir.
- İki farklı sayı türü vardır!
  - (5, 4, 3, 6) tam sayılardır - onlarda kesir yoktur
  - (.25, .10, .05, .01) ondalık kesirlerdir
  - Bilgisayarın içinde tam sayılar ve ondalık kesirler ikili sayı tabanında oldukça farklı şekilde temsil edilir!
  - Ondalık kesirler ve tam sayıların iki farklı veri türü olduğunu söylüyoruz.
- Bir nesnenin veri türü, hangi değerlere sahip olabileceğini ve üzerinde hangi işlemlerin gerçekleştirilebileceğini belirler.
- Tam sayılar, tamsayı (kısaca int) veri türü kullanılarak temsil edilir. Bu değerler pozitif veya negatif tam sayılar olabilir.
- Kesirli kısımlara sahip olabilen sayılar, kayan nokta (veya kayan nokta) değerleri olarak temsil edilir.
- Hangisinin hangisi olduğunu nasıl anlarız?
- Ondalık noktası olmayan sayısal sabit değer, bir int değeri üretir
- Ondalık noktası olan bir sabit değer, kayan nokta ile temsil edilir (kesirli kısım 0 olsa bile)

Sayısal Veri Türleri

- Python'un bize herhangi bir değerin veri türünü söyleyen özel bir işlevi vardır.

a=type(3)

print(a)

<class 'int'>

a=type(‘Ahmet’)

print(a)

<class 'str'>

type(3.1)

<class 'float'>

type(3.0)

<class 'float'>

myInt = 32

type(myInt)

<class 'int'>

b=25

a=type(b)

print(a)

Sayısal Veri Türleri

- Neden iki sayı türüne ihtiyacımız var?
  - Sayıları temsil eden değerler kesirli olamaz (3 ½ çeyreğe sahip olamazsınız)
  - Çoğu matematiksel algoritma tam sayılarla çok verimlidir
  - Float türü, temsil edilen gerçek sayıya yalnızca bir yaklaşıklık depolar!
  - Değişkenler kesin olmadığından, mümkün olduğunda bir int kullanın!

Sayısal Veri Türleri

- int'ler üzerindeki işlemler int'leri üretir, float'lar üzerindeki işlemler float'ları üretir (/ dışında).

```
3.0+4.0
7.0
3+4
7
3.0*4.0
12.0
3*4
12
10.0/3.0
3.3333333333333335
10/3
3.3333333333333335
10 // 3
3
10.0 // 3.0
3.0
```

```
b=25.2 + 12.8
a=type(b)
print(b)
print(a)
38.0
<class 'float'>

b=25 + 13
a=type(b)
print(b)
print(a)
38
<class 'int'>

b=(8 + 2)/3
a=type(b)
print(b)
print(a)
3.3333333333333335
<class 'float'>
```

Sayısal Veri Türleri

- Tamsayı bölme bir tam sayı üretir.
- Bu yüzden 10//3 = 3
- Bunu ‘tamsayısı' olarak düşünün, burada 10//3 = 3 çünkü 3 *3=9 (kalan 1 olacak)
- 10%3 = 1, 10'un 3'e tam bölümünden kalandır.
- a = (a/b)(b) + (a%b)

Matematik Kütüphanesini Kullanma

- (+, -, *, /, //, **, %, abs) dışında, bir matematik kütüphanesinde birçok başka matematik fonksiyonumuz mevcuttur.
- Bir kitaplık, bazı yararlı tanımlara/işlevlere sahip bir modüldür.
- İkinci dereceden bir denklemin köklerini hesaplayan bir program yazalım!

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Bunun nasıl yapılacağını bilmediğimiz tek kısmı karekök bulmak... ama o matematik kütüphanesinde!

Matematik Kütüphanesini Kullanma

- Matematik kütüphaneyi kullanmak için şu satırın programımızda olduğundan emin olmamız gerekir: **import math**
- Matematik kütüphanenin içe aktarılması, içinde tanımlanan matematiksel işlevlerin yazılan program tarafından kullanılabilir olmasını sağlar.
- sqrt kütüphanesi rutinine erişmek için, ona math.sqrt(x) olarak erişmemiz gerekir.
- Bu nokta gösterimini kullanmak, Python'a matematik kütüphane modülünde bulunan sqrt işlevini kullanmasını söyler.
- Kökü hesaplamak için discRoot = math.sqrt(b*b – 4*a*c) yazabilirsiniz.

```
import math
a=1.5
b=5
c=1.5
e=b*b-4*a*c
d=math.sqrt(e)
print(d)
```

```
4
```

Matematik Kütüphanesi

- from math import sin, pi
- print('Pi is roughly', pi)
- print('sin(0) =', sin(0))
- Pi is roughly 3.14159265359
- sin(0) = 0.0
- Built-in math functions There are two built in math functions, abs (absolute value) and round
- that are available without importing the math module. Here are some examples:
- print(abs(-4.3))
- print(round(3.336, 2))
- print(round(345.2, -1))
- 4.3
- 3.37
- 350.0
- The round function takes two arguments: the first is the number to be rounded and the second is
- the number of decimal places to round to. The second argument can be negative.

Matematik Kütüphanesini Kullanma

```
# quadratic.py
#    A program that computes the real roots of a quadratic equation.
#    Illustrates use of the math library.
#    Note: This program crashes if the equation has no real roots.

import math  # Makes the math library available.

def main():
    print("This program finds the real solutions to a quadratic")
    print(“----------------------------------------------------------------”)

    a, b, c = eval(input("Please enter the coefficients (a, b, c): "))

    discRoot = math.sqrt(b * b - 4 * a * c)
    root1 = (-b + discRoot) / (2 * a)
    root2 = (-b - discRoot) / (2 * a)

    print()
    print("The solutions are:", root1, root2 )

main()
```

Matematik Kütüphanesini Kullanma

```
This program finds the real solutions to a quadratic

Please enter the coefficients (a, b, c): 3, 4, -1

The solutions are: 0.215250437022 -1.54858377035
```

- What do you suppose this means?

```
This program finds the real solutions to a quadratic

Please enter the coefficients (a, b, c): 1, 2, 3

Traceback (most recent call last):
  File "<pyshell#26>", line 1, in -toplevel-
    main()
  File "C:\Documents and Settings\Terry\My Documents\Teaching\W04\CS 120\Textbook\code\chapter3\quadratic.py", line 14, in main
    discRoot = math.sqrt(b * b - 4 * a * c)
ValueError: math domain error
>>>
```

Matematik Kütüphanesi

- a = 1, b = 2, c = 3 ise, negatif bir sayının karekökünü almaya çalışıyoruz!
- sqrt işlevini kullanmak, ** kullanmaktan daha verimlidir.
- Karekökü hesaplamak için **'ı nasıl kullanırsınız?

Birikimli Sonuçlar: Faktöryel

- Diğer beş kişiyle birlikte sırada beklediğinizi varsayalım.
- Altı kişiyi düzenlemenin kaç yolu var?
- 720 -- 720, 6'nın faktöriyelidir (kısaltılmış 6!)
- Faktöriyel şu şekilde tanımlanır: n! = n(n-1)(n-2)…(1)
- Yani, 6! = 6*5*4*3*2*1 = 720

Birikimli Sonuçlar: Faktöryel

- Bunu yapmak için nasıl bir program yazabiliriz?
- Faktöriyeli alınacak girdi sayısı, n n'nin faktöriyelini hesapla, gerçek
- Çıktı olgusu

6! Nasıl hesaplanır?

- 6*5 = 30
- 30 alınır, ve 30 * 4 = 120
- 120 alınır, ve 120 * 3 = 360
- 360 alınır, ve 360 * 2 = 720
- 720 alınır, ve 720 * 1 = 720

Birikimli Sonuçlar: Faktöryel

- Gerçekten neler oluyor?
- Tekrarlanan çarpmalar yapıyoruz ve çalışan çarpımı takip ediyoruz.
- Bu algoritma akümülatör olarak bilinir, çünkü cevabı akümülatör değişkeni olarak bilinen bir değişkende oluşturur veya biriktiririz.
- Bir akümülatör algoritmasının genel biçimi şöyle görünür:
- Akümülatör değişkenini başlat
- Nihai sonuca ulaşılana kadar döngü biriktirici değişkenin değerini güncelleyin

Birikimli Sonuçlar: Faktöryel

- Bir döngüye ihtiyacımız olacak gibi görünüyor!

```
fact = 1
for factor in [6, 5, 4, 3, 2, 1]:
    fact = fact * factor
```

- Bunun işe yaradığını doğrulamak için izini sürelim!
- Neden fact'i 1 olarak başlatmamız gerekti?
- Birkaç sebep var…
- Döngü boyunca her seferinde, bir sonraki gerçeğin değerini hesaplamak için bir önceki gerçeğin değeri kullanılır.
- Başlatmayı yaparak, gerçeğin ilk seferinde bir değere sahip olacağını bilirsiniz.
- Bir değer atamadan gerçeği kullanırsanız, Python ne yapar?

Birikimli Sonuçlar: Faktöryel

- Çarpma birleştirici ve değişmeli olduğundan, programımızı şu şekilde yeniden yazabiliriz:

```
fact = 1
for factor in [2, 3, 4, 5, 6]:
    fact = fact * factor
```

- Harika! Peki ya başka bir sayının faktöriyelini bulmak istiyorsak?

```
fact = 1
for factor in [2, 3, 4, 5, 6]:
    fact = fact * factor
    print(fact)
```

```
2
6
24
120
720
```

```
fact = 1
for factor in [2, 3, 4, 5, 6]:
    fact = fact * factor

print(fact)
```

```
720
```

Birikimli Sonuçlar: Faktöryel

- range(n) ne döndürür?
  0, 1, 2, 3, …, n-1
- Range komutunun başka bir isteğe bağlı parametresi vardır!
- *range(start, n)*
  start, start + 1, …, n-1
- Fakat bekleyin! Fazlası var!
  range(start, n, step)
  start, start+step, …, n-1
- list(<sequence>) bir liste yapar

Bazı örnekler deneyelim!

list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

list(range(5,10))
[5, 6, 7, 8, 9]

list(range(5,10,2))
[5, 7, 9]

Birikimli Sonuçlar: Faktöryel

- Güçlendirilmiş range ifadesini kullanarak, döngümüz için aralığı birkaç farklı şekilde yapabiliriz.
  - 2'den n'ye kadar sayabiliriz:
  - range(2, n+1)
    (Neden n+1 kullanmak zorundaydık??)
  - n'den 2'ye kadar geri sayabiliriz:
    range(n, 1, -1)

Birikimli Sonuçlar: Faktöryel

- Tamamlanan faktoriyel program:

```
# factorial.py
#    Program to compute the factorial of a number
#    Illustrates for loop with an accumulator

def main():
    n = eval(input("Please enter a whole number: "))
    fact = 1
    for factor in range(n,1,-1):
        fact = fact * factor
    print("The factorial of", n, "is", fact)

main()
```

Int'nin Sınırları

- 100!?

>>> main()

Please enter a whole number: 100

The factorial of 100 is
933262154439441526816992388562667004907159682643816214685929
638952175999932299156089414639761565182862536979208272237582
511852109168640000000000000000000000

- Vay! Bu oldukça büyük bir rakam!

Int'nin Sınırları

- Python'un daha yeni sürümleri bununla başa çıkabilir, ancak…

```
Python 1.5.2 (#0, Apr 13 1999, 10:51:12) [MSC 32 bit (Intel)] on win32
Copyright 1991-1995 Stichting Mathematisch Centrum, Amsterdam
>>> import fact
>>> fact.main()
Please enter a whole number: 13
13
12
11
10
9
8
7
6
5
4
Traceback (innermost last):
  File "<pyshell#1>", line 1, in ?
    fact.main()
  File "C:\PROGRA~1\PYTHON~1.2\fact.py", line 5, in main
    fact=fact*factor
OverflowError: integer multiplication
```

Int'nin Sınırları

- Neler oluyor?
  - Sonsuz sayıda tamsayı varken, temsil edilebilecek sonlu bir int aralığı vardır.
  - Bu aralık, belirli bir CPU'nun bir tamsayı değerini temsil etmek için kullandığı bit sayısına bağlıdır.
- Tipik PC'ler 32 bit kullanır
- Bu, 0 merkezli 2^32 olası değer olduğu anlamına gelir.
- Bu aralık -2^31 ila 2^31-1 arasındadır. 0'ı hesaba katmak için üst uçtan bir çıkarmamız gerekiyor.
- Ama bizim 100! bundan çok daha büyük. O nasıl çalışır?

Büyük Sayıları İşleme

- Kayan veri türlerine geçmek bizi int'lerin sınırlamalarından kurtarıyor mu?Akümülatörü 1.0 olarak başlatırsak,

```
>>> main()
Please enter a whole number: 15
The factorial of 15 is 1.307674368e+012
```

- Artık kesin bir cevap alamıyoruz!
- Çok büyük ve çok küçük sayılar bilimsel veya üstel gösterimle ifade edilir.
- 1,307674368e+012, 1,307674368 * 10^12 anlamına gelir
- Burada orijinal sayıyı elde etmek için ondalık basamağın 12 ondalık basamak sağa taşınması gerekir, ancak yalnızca 9 basamak vardır, bu nedenle 3 basamak kesinlik kaybedilmiştir.

Büyük Sayıları İşleme

- Floats yaklaşık değerlerdir
- Floats, daha geniş bir değer aralığını, ancak daha düşük hassasiyetle temsil etmemizi sağlar.
- Python'un Tns‘leri genişleten bir çözümü var!
- Python Ints sabit bir boyut değildir ve sahip olduğu değer ne olursa olsun işlemek için genişler.
- Python'un daha yeni sürümleri, girişlerinizi taşacak kadar büyüdüklerinde otomatik olarak genişletilmiş forma dönüştürür
- .Hız ve bellek pahasına sonsuz büyük değerler (ör. 100!) alıyoruz

Dönüşüm Tipleri

- Bir int ile bir int'yi birleştirmenin bir int ürettiğini ve bir float ile bir float'ı birleştirmenin bir float ürettiğini biliyoruz.
- Bir ifadede bir int ve float'ı karıştırdığınızda ne olur?
  x = 5.0 + 2
- Python'un bu ifadeyi değerlendirebilmesi için ya 5.0'ı 5'e dönüştürüp bir tam sayı toplaması yapması ya da 2'yi 2.0'a dönüştürmesi ve bir kayan nokta toplaması yapması gerekir.
- Bir kayan noktayı int'ye dönüştürmek bilgi kaybına neden olur Ints, ".0" eklenerek değişkenlere dönüştürülebilir.
- Karışık türde ifadelerde Python, int'leri değişkenlere dönüştürür.Bazen tür dönüştürmeyi kontrol etmek isteriz. Buna açık yazma denir.

Dönüşüm Tipleri

```
>>> float(22//5)
4.0
>>> int(4.5)
4
>>> int(3.9)
3
>>> round(3.9)
4
>>> round(3)
3
```