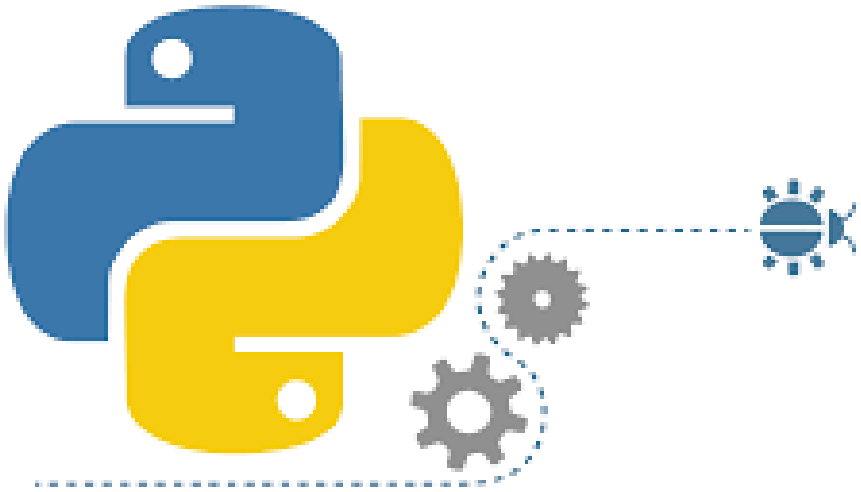
PYTHON
Programalamaya Giriş



Öğr. Gör. Koray AKİ

Python Nasıl Bir Dildir?

- Yüksek Seviyeli
- Genel Amaçlı Prosedürel
- Fonksiyonel
- Nesne Yönelimli Programlama Modellerini Destekleyen Çok Modelli (Multiparadigm)
- Dinamik tür sistemine (dynamic type system) sahip bir programlama dilidir.

Python Nasıl Bir Dildir?

- Python ilk olarak **Guido Van Rossum** tarafından 1990'lı yılların başlarında geliştirilmeye başlanmıştır. Ortaya çıkışında bir başka çok amaçlı programlama dili olan ABC'nin etkisi olmuştur.
- Python'un ilk sürümü olan 0.9.0, **20 Şubat 1991** tarihinde kullanılmaya başlanmıştır.
- Python'un günümüzde en çok kullanılan ve en stabil sürümleri 2.7 ve 3.7 sürümleridir.
- Zannedildiğinin aksine bu programlama dilinin adı piton yılanından gelmez... Guido Van Rossum bu programlama dilini, "The Monty Python" adlı bir İngiliz komedi grubunun, "Monty Python's Flying Circus" adlı gösterisinden esinlenerek adlandırmıştır.

python

Python Dilinin Tarihsel Gelişimi

- ✓ Aralık 1989 Tasarım ve gerçekleştirime başlandı
- ✓ 1990 İlk versiyonlara içsel numnaralar verilmiştir
- ✓ Şubat 1991 0.9
- ✓ Ocak 1994 1.0
- ✓ Ekim 1994 1.1
- ✓ Ekim 1995 1.3
- ✓ ---
- ✓ Aralık 2023 3.12.1
- ✓ Şubat 2024 3.12.2
- ✓ ---
- ✓ Aralık 2024 3.13.1

https://www.python.org/doc/versions/

Python Programlama Dilinin geliştirilmesi 2001'den bu yana "***Python Software Foundation***" isimli kurum tarafından yapılmaktadır (www.python.org).

Python Programlama Dili

- Python standardizasyon komiteleri tarafından resmi olarak standardize edilmiş bir dil değildir. Dilin sentaks ve semantik özellikleri "***Python Language Reference***" isimli dokümanlarda tanımlanmıştır (https://docs.python.org/3/reference/index.html).
- Python dünyasında "gerçekleştirim (implementation)" denildiğinde yazılmış çalıştırılabilen Python yorumlayıcıları ve kütüphaneleri anlaşılmaktadır.
- Python dilinin pek çok gerçekleştirimi (implementation) vardır.

Python Gerçekleştirimleri

- Python'ın en önemli ve en çok kullanılan gerçekleştirimi "**Python Software Foundation**" tarafından sürdürülen (ilk versiyonları Guido van Rossum tarafından yazılmıştır) olan **CPython** isimli yazılımdır.
- **Jython** isimli yorumlayıcı sistem Java'da yazılmıştır. Jython'da yazılmış olan programlar Java sınıflarını doğrudan kullanabilirler.
- **IronPython** gerçekleştirimi Jthon gerçekleştiriminin .NET (CLI) versiyonu gibidir. IronPython C# Programlama Dili kullanılarak yazılmıştır.
- **PyPy** önemli Python gerçekleştirimlerinden biridir. PyPy üretilen ara kodu yorumlayıcı olartak değil JIT Derlemesi (Just in Time Compilation) yaparak çalıştırır. Bu yüzden PyPy standard Python gerçekleştirimi olan CPython'dan daha yüksek bir çalışma zamanı performansına sahiptir.

https://wiki.python.org/moin/PythonImplementations

Python Dağıtımları

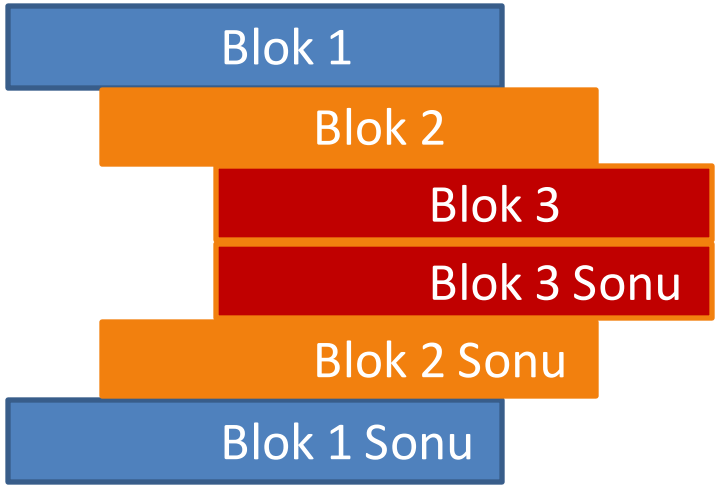
- Python dağıtımları belli bir Python gerçekleştirimi temel alınıp onlara çeşitli araçlar eklenerek oluşturulmuş paketlerdir
- Python dağıtımları hem Python yorumlayıcılarını hem de uygulama geliştirmeye yardımcı olabilecek birtakım araçları barındırmaktadır.
- CPython dağıtımı kendi içerisinde CPython yorumlayıcı sistemini, çeşitli kütüphaneleri ve **IDLE** (Integrated Development and Learning Environment) isimli basit bir IDE'yi barındırmaktadır.
- Continuum Analytics firması tarafından açık kaynak kodlu (dolayısıyla da ücretsiz) olarak dağıtılan Anaconda CPython'dan sonra en çok kullanılan dağıtımlardan biridir. Anaconda yorumlayıcı sistem olarak CPyton kullanmaktadır.
- ActiveState firması (bu firma aynı zamanda Komodo IDE'sinin üreticisidir) tarafından geliştirilmiş ActivePython dağıtımı en yaygın kullanılan dağıtımlardan biridir. ActivePython dağıtımının ücretli ve "Community Edition" ismiyle ücretsiz sürümleri de vardır. ActivePython yorumlayıcı sistem olarak CPython gerçekleştirimini kullanmaktadır.

https://wiki.python.org/moin/PythonDistributions

Sentaks Gösterimleri

- Programlama dillerinin sentakslarını betimlemek için pek çok meta dil oluşturulmuştur. Bunlardan en ünlüsü ve çok kullanılanı **BNF (Backus-Naur Form)** denilen notasyondur.
- Python'ın orijinal referans kitabında da bu notasyon tercih edilmiştir.
- **BNF** notasyonu **ISO** tarafından da standardize edilmiştir. İsmine **EBNF** denilmektedir.

```
if (<ifade>)
    <deyim>
[
else
    <deyim>
]
```



Veri Tipleri

Tür İsmi	Özelliği
int	İşaretli tamsayı türü. Python'da int türünün bir sınırı yoktur.
float	IEEE 754 Standardının 8 byte'lık gerçek sayı formatını (long real format) belirtmektedir.
bool	Bu tür True ya da False biçiminde belirtilen ikil bilgileri tutmaktadır.
str	Yazıları tutan string türüdür. Yazılar tek tırnakla ya da iki tırnakla belirtilebilirler.
NoneType	Python'da özel bir NoneType türü vardır. Bu tür None anahtar sözcüğü ile temsil edilmektedir. None boş değer anlamına gelir.
complex	Python'da karmaşık sayı türü de vardır. i sayısı j ile temsil edilmektedir.

Python'da Yorumlama

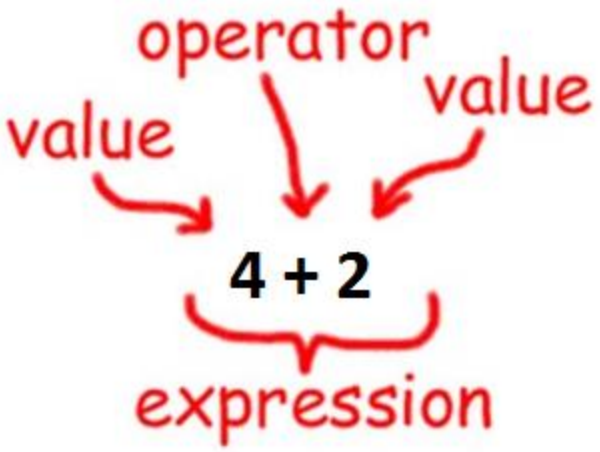
- Python'da satır sonuna kadar yorumlama '#' ile yapılmaktadır.
- Python'da eğer üç tek tırnaktan ya da üç çift tırnaktan sonra yeni satıra geçilmişse bir dahaki üç tek tırnağa ya da üç çift tırnağa kadarki bölge yorumlayıcı tarafından ele alınmaz.

Örneğin:

```
# Test programı
"""
Bu satırlar
yorumlayıcı tarafından
dikkate alınmaz
"""
```

Operatörler (İşlemler)

- 4 + 2 işlemi 6 olarak hesaplanır
- 4 ve 2 operant (işlenen, işleneç); + ise operatördür



Operatörler – 1/3

İşlem	Tanım	Örnek
+	Toplama	a + b
-	Çıkarma	a-b
*	Çarpma	a*b
/	Bölme	b/a
%	Mod alma işlemi (bölmede kalan)	b%a

Operatörler – 2/3

İşlem	Tanım	Örnek
**	Üs alma işlemi	2**4
//	Taban bölmesi (bölümü hesaplar). Eğer operantlar tamsayı ise sonuç tamsayıdır; eğer operantların en az biri reel sayı ile sonuç reel sayıdır ama kesir kısmı 0'dır.	9//2 ve 9.0//2.0
==	İki değer birbirine eşit mi diye kontrol eder (sonuç TRUE veya FALSE olur)	(a==b)
!=	İki değer birbirinden farklı mı diye kontrol eder (sonuç TRUE veya FALSE olur)	(a!=b)
<>	İki değer birbirinden farklı mı diye kontrol eder (sonuç TRUE veya FALSE olur)	(a<>b)

Operatörler – 3/3

İşlem	Tanım	Örnek
>	Büyük mü karşılaştırması (sonuç TRUE veya FALSE olur)	(a>b)
<	Küçük mü karşılaştırması (sonuç TRUE veya FALSE olur)	(a<b)
>=	Büyük ya da eşit mi karşılaştırması (sonuç TRUE veya FALSE olur)	(a>=b)
<=	Küçük ya da eşit mi karşılaştırması (sonuç TRUE veya FALSE olur)	(a<=b)

http://www.tutorialspoint.com/python/python_basic_operators.htm

Değişkenler

- Diğer dillerde de olduğu gibi bir değişken **RAKAM** ile **başlayamaz**.
- İlk karakter bir harf veya _ (altçizgi) olmak zorundadır.
- Harf, Rakam ve _ (alt çizgi) haricinde bir karakter **içeremez**. (**ÖR:** $, #,*, ? veya boşluk gibi).
- Aksi belirtilmedikçe tüm değişkenler yerel olarak algılanırlar.

String Değişkenler

- String bir değişkene değer atamak için “ (çift tırnak) veya ‘ (tek tırnak) ifadesi kullanılır. Eğer karakter dizisi belirtilirken çift tırnak kullanılırsa o karakter dizisi içerisinde çeşitli özel karakterler (%s , %d vs.) aranır , varsa değiştirilir.

  - **>>>ad=“Koray”**
  - **>>>ad=‘Koray’**

    - ✓ **>>>=a=9**
    - ✓ **>>>b=“5”**
    - ✓ **>>>c=a+b**
    - ✓ **Traceback (most recent call last):**
    - ✓ **File "<pyshell#11>", line 1, in <module>**
    - ✓ **a+b**
    - ✓ **TypeError: unsupported operand type(s) for +: 'int' and 'str‘**

Üs Alma ve Mod İşlemleri - Örnekler

```
>>> a=3
>>> b=2
>>> a**b
9
>>> 5**2
25
>>> 25**0.5
5.0
>>> 5**3
125
>>> 5**4
625
```

```
>>> 16%5
1
>>> 18%4
2
>>> 30%2
0
```

İşlem ve Atamanın bir arada yapılması

c += a

aslında **c = c + a** demektir

c -= a

aslında **c = c - a** demektir

c *= a

aslında **c = c * a** demektir

c /= a

aslında **c = c / a** demektir

Aslında diğer operatörler de bu mantıkla kullanılabilir

Karşılaştırma İşlemleri

Eğer operantlar eşit ise True
Aksi halde False

```
>>> 3==4
False
>>> 3==3
True
>>> "python"=="pon"
False
>>> "python"=="python"
True
```

Eğer operantlar eşit değil ise True
Aksi halde False

```
>>> 3!=4
True
>>> 3!=3
False
>>> "python"!="pon"
True
>>> "python"!="python"
False
```

Karşılaştırma İşlemleri

a > b

Eğer a, b'den büyükse True
Aksi halde False

a >= b

Eğer a, b'den büyük ya da eşit ise True
Aksi halde False

a < b

Eğer a, b'den küçükse True
Aksi halde False

a <= b

Eğer a, b'den küçük ya da eşit ise True
Aksi halde False

```
>>> 5>4
True
>>> 4>4
False
>>> 4>=4
True
>>> 3<4
True
>>> 4<4
False
>>> 4<=4
True
```

cTrue

Karşılaştırma İşlemleri

```
>>> 10 + 2 > 3 + 8
True

>>> a = 10
>>> 5 < a < 20
True
İ1: 5 < a
İ2: İ1 < 20

>>> a = 10
```

```
>>> a =10
>>> 10 == a > 5
True
```

Buradaki işlemin eşdeğeri şöyledir:

```
10 == a and a > 5

>>> 3 < a < 20 > 10
True
```

İşleminin eşdeğeri de şöyledir:

```
3 < a and a < 20 and
20 > 10
```

```
>>> a =10
>>> b =10
>>> c =10
>>> a == b == c
True
```

Buradaki işlemin eşdeğeri şöyledir:

```
a == b and b == c
```

Örnekler

- Girilen bir sayının tek mi çift mi olduğunu bulan bir program yazalım:

```
sayi = int(raw_input("Bir sayi giriniz: "))
if sayi%2 == 0:
    print "Girdiginiz sayi %d, bir cift tam sayidir" %sayi
else:
    print "Girdiginiz sayi %d, bir tek tam sayidir" %sayi
```

```
>>> Bir sayi giriniz: 5
Girdiginiz sayi 5, bir tek tam sayidir
>>> Bir sayi giriniz: 20
Girdiginiz sayi 20, bir cift tam sayidir
>>>
```

İşlemlerin Öncelikleri

İşlem	Tanımı
()	Parantez, işlemleri gruplar
**	Üs alma işlemi
+ -	İşaret Operatörü
* / // %	Çarpma, bölme, taban bölme, kalan
+ -	Toplama, çıkarma
< <= > >=	Karşılaştırmalar
<> != ==	Eşitlik
not	Mantıksal operatörler
and	Mantıksal operatörler
or	Mantıksal operatörler
= += -= *= ve diğer atamalar	Atamalar

İşlemlerin Öncelikleri

a = 20, b = 10, c = 15, d = 5, e = 0

```
e = (a + b) * c / d
print "Value of (a + b) * c / d is ",  e

e = ((a + b) * c) / d
print "Value of ((a + b) * c) / d is ",  e

e = (a + b) * (c / d);
print "Value of (a + b) * (c / d) is ",  e

e = a + (b * c) / d;
print "Value of a + (b * c) / d is ",  e
```

```
>>>
Value of (a + b) * c / d is  90
Value of ((a + b) * c) / d is  90
Value of (a + b) * (c / d) is  90
Value of a + (b * c) / d is  50
>>>
```

İşlemlerin Öncelikleri

- Aynı seviyedeki işlemlerde öncelik genelde soldan-sağa doğrudur

```
>>> 5*2//3
3
>>> 5 * (2 // 3)
0
```

- İstisnai durum (** için sağdan sola doğrudur)

```
>>> 2 ** 3 ** 2 ,
512
>>> (2**3)**2
64
```

- Diğer bir istisnai durum (çoklu karşılaştırmalar matematikteki gibidir)

```
>>> 10 > 6 > 2
True
>>> 5 < 3 < 6
False
```

String İşlemleri

- Bir “string” değişkeninin uzunluğunu bulmak isteyebiliriz.
- Bunun için “len()” fonksiyonunu kullanırız
- Örnek:

```
>>> cumle = «Uludağ Üniversitesi'ne hos geldiniz!"
>>> U = len(cumle)
>>> print "Cumlenin uzunlugu %d karakterdir" %U
Cumlenin uzunlugu 37 karakterdir
>>>
```

print Fonksiyonu

```
>>> a = 10
>>> b = 20
>>> c = 30
>>> print(a, b, c)
10 20 30
>>> print(a * 2, b * 3, c * 4)
20 60 120


>>> print('a =', a)
a = 10
```

print default olarak argümanlar arasına bir tane SPACE karakteri basmaktadır. Ancak bu karakter sep isimli parametre belirtilerek değiştirilebilir.

```
>>> a = 10; b = 20; c = 30
>>> print(a, b, c, sep =',')
10,20,30


>>> print(a, b, c, sep='xxx')
10xxx20xxx30
>>> print(a, b, c, sep ='')
102030
```

print Fonksiyonu

print en son argümanı da yazdırdıktan sonra default olarak imleci aşağıdaki satrın başına geçirir. Ancak bu da end parametresiyle ayarlanabilmektedir.

```
>>> print(a, b, c, sep ='', end='.’)
102030.

>>> print(1); print(2); print(3)
123
>>> print(1, end=' ');print(2,  end=' ‘); print(3)
1 2 3
```

input Fonksiyonu

Python'da klavyeden yazı okumak için input isimli fonksiyon kullanılmaktadır. input fonksiyonu parametresiyle aldığı yazıyı ekrana basar ve klavyeden bir yazı bekler. Kullanıcı yazıyı yazıp ENTER tuşuna bastığında input yazılan yazıyı bir string nesnesine yerleştirir ve o stringi bize verir.

```
>>> s = input('Bir isim giriniz:’)
Bir isim giriniz:Koray
>>> s
‘Koray’
```

```
>>> n = int(input('Bir sayı giriniz:')); print(n * n)
Bir sayı giriniz:10
100
```

String İçindeki Karakterlere Erişmek

- String içindeki karakterlerin indeksleri vardır. İndeksler 0'dan başlar (sol baş karakterin indeksi) ve boyunun bir eksiğine kadar gider (sağdaki son karakterin indeksi)
- Örnek:

```
>>> isim = "koray"
            0 1 2 3 4
```

```
>>> print (len(isim))
5
>>> print (isim[4])
y
```

String İçindeki Karakterlere Erişmek

```
>>> isim = “Koray Aki"
>>> print (isim[6:9])
Aki
>>> print (isim[4:8])
y Ak
>>> print (isim[1:9:2])
oa k
>>> print (isim[8:3:-1])
ikA y
>>> print (isim[:])
Koray Aki
>>> print (isim[::-1])
ikA yaroK
```

String Üzerinde İşlemler

```
>>> isim = “koray aki"
>>> print(isim)
koray aki
>>> print (isim.capitalize())
Koray aki
>>> print (isim.upper())
KORAY AKİ
>>> print (isim.title())
Koray Aki
```

Stringleri Birleştirmek

- Karakter dizilerinin “+” ya da “,” işaretleriyle birleştirebiliriz
- Örnek:

```
>>> print("Uludağ" + "Üniversitesi")
UludağÜniversitesi
>>> print("Uludağ”, "Üniversitesi")
Uludağ Üniversitesi
>>>
```

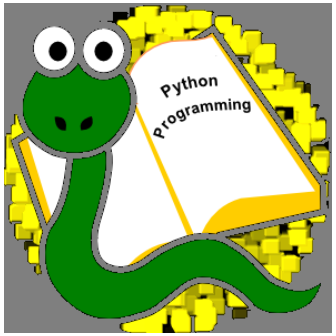
Aradaki fark nedir?

Arada boşluk olup olmaması

String Üzerindeki İşlemler

- String değişkenler üzerinde farklı işlemler yapabiliriz
- Örnek:

```
>>> dil = "python"
>>> print dil*10
pythonpythonpythonpythonpythonpythonpythonpythonpythonpython
```



Örnekler

```
>>> "ton" *2
'tonton'
>>> "x"*30
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
>>> word="-"
>>> word*30
'------------------------------'
```

Sınıf içi Çalışma

- Ekrana alt ve üst kenarları “-” ile, yan kenarları “|” ile oluşturulan, uzunluğu ve yüksekliği kullanıcı tarafından girilen bir dikdörtgen çizebilir misiniz?
- Örnek: Uzunluk 10, yükseklik 5 birim ise çizilecek şekil aşağıdaki gibi olur

```
----------
|        |
|        |
|        |
----------
```

Çözüm

```
yan_kenar = '|'
cizgi = '-'
bosluk = ' '

a = int(raw_input("uzunlugu girin: "))      # uzunluk
b = int(raw_input("yuksekligi girin: "))    # yukseklik

iki_kenar = yan_kenar + (a-2)*bosluk + yan_kenar

print a*cizgi
for i in range(1,b-1):
    print iki_kenar
print a*cizgi
```

Üyelik (Membership) İşlemleri

```
>>> word="hello"
>>> "h" in word
True
>>> "k" in word
False
>>> "h" not in word
False
>>> "k" not in word
True
>>> "lo" in word
True
>>> "elo" not in word
True
```

in operatörü üyelik durumunu sorgular ve True ya da False döndürür

not in operatörü tersini yapar

Bugünün Tarihini Nasıl Buluruz?

- Bunun için Python'da hazır yazılmış modüller var.
- Modül bunu ve benzer diğer fonksiyonları kullanmanızı sağlar
- Modül isminden önce import anahtar kelimesini yazarız. Böylece o modülü program içinde kullanabiliriz
- Örneğin "datetime" isimli modülü kullanmak için programın başında aşağıdakini yazmamız gerekir.

```
import datetime
```

Örnek kod:

```
import datetime

t = datetime.date.today()

print t
print t.day
```

```
>>>
2017-04-28
28
>>>
```

Diğer Tarih İşlemleri - Detaylar

```
import datetime
t = datetime.date.today()
```

t bir tarih değişkeni; üzerinde fonksiyonlar çalışabilir

```
gun = t.day
ay = t.month
yil =  t.year
```

year yılı, month ayı, day ayın kaçıncı günü olduğunu integer olarak verir

```
t2 = datetime.datetime.today()
print t2
```

t2 de bir tarih değişkeni ama saati de tutuyor. Çıktıyı deneyip görelim

Diğer Tarih İşlemleri – Değişik Formatlarda Yazdırma

```
import datetime
t = datetime.date.today()

print ("Bugunun tarihi:", t.strftime("%d-%m-%Y"))
print ("Bugunun tarihi:", t.strftime("%m/%d/%y"))
print ("Bugunun tarihi:", t.strftime("%d.%m.%Y"))
```

```
Bugunun tarihi: 27-04-2017
Bugunun tarihi: 04/27/17
Bugunun tarihi: 27.04.2017
```

strftime fonksiyonu tarihi istediğiniz formatta stringe çevirir. Formatı çift tırnak içinde istediğiniz ayraçlarla ve istediğiniz gün (%d), ay(%m) ve yıl (%y veya %Y) sırasında belirtebilirsiniz

Diğer Tarih İşlemleri – Klavyeden Tarih Girme

- Klavyeden tarihi girmek için önce istediğiniz formatta string olarak okumanız gerekir.
- Sonra datetime.strptime fonksiyonunu kullanarak string'i tarihe cevirip tarih değişkeni içinde saklarız.
  - İlk parametre okuduğumuz tarih string'i, ikinci parametre ise format string'i
- Öncesinde `from datetime import datetime`
- Örnek:

```
from datetime import datetime

tarih_string = raw_input("Istediginiz tarihi GG-AA-YYYY seklinde giriniz: ")
girilen = datetime.strptime(tarih_string, "%d-%m-%Y")
print "Girilen tarih:", girilen.strftime("%m.%d.%Y")
```

```
Istediginiz tarihi GG-AA-YYYY seklinde giriniz: 17-07-2001
Girilen tarih: 07.17.2001
```

Diğer Tarih İşlemleri – İki Tarih Arasındaki Farkı Bulma

- t1 ve t2 iki tarih (datetime) değişkeni olsun.
- t1 – t2 ifadesi timedelta denilen bir tipten ifadedir.
- Bunun üzerinde days fonksiyonunu çağırarak aradaki gün sayısını bulabiliriz.

```
fark = t1 – t2
print (fark.days)
```

Örnek

- Doğum yılını okuyup kaç yaşında olduğunu bulan bir program yazalım

```
import datetime
dy = int(raw_input("Dogum yilinizi giriniz: "))
t = datetime.datetime.today()
yas =  t.year-dy
print "%d yasindasin" %yas
```

```
>>>
Dogum yilinizi giriniz: 1990
27 yasindasin
>>>
```

http://www.tutorialspoint.com/python/python_date_time.htm

Ödev

- Programınız sizden doğum tarihinizi gün-ay-yıl olarak girmenizi isteyecek (yıl 4 hane)
- Önce bugünün tarihini gün-ay-yıl olarak bastıracak (yıl 4 hane)
- Sonra girilen doğum gününü gün-ay-yıl olarak bastıracak (yıl 4 hane)
- Daha sonra sizin toplam kaç gündür yaşadığınızı hesaplayıp ekrana bastıracak.
- Eğer girilen doğum günü bugünden sonra ise hata mesajı yazsın.
- Eğer girilen doğum günü bugün ise bugün doğulduğunu belirten bir mesaj yazsın

Örnek Çıktı 1

```
>>>

Dogum tarihinizi, GG-AA-YYYY seklinde, giriniz: 17-07-2001

Bugunun tarihi: 28-04-2017

Dogum tarihi: 17-07-2001

5764 gundur yasiyorsunuz

>>>
```

Örnek Çıktı 2

```
>>>

Dogum tarihinizi, GG-AA-YYYY seklinde, giriniz: 22-06-2017

Bugunun tarihi: 28-04-2017

Dogum tarihi: 22-06-2017

Daha dogmadiniz ki!!!

>>>
```

Örnek Çıktı 3

```
>>>

Dogum tarihinizi, GG-AA-YYYY seklinde, giriniz: 28-04-2017

Bugunun tarihi: 28-04-2017

Dogum tarihi: 28-04-2017

Bugun dogdunuz. Dunyaya hos geldiniz!!!

>>>
```