

Python'a Bařlangıç

Bu belgede çok güzel bir dil olan python programlama diline kısa bir giriş yapacağız.
ve sorularınızı yanıtlamaya çalışacağız.

Öncelikle python dan biraz bahsedelim,

python çok kolay ve kullanışlı bir dildir, sistem programcılığı için çok uygundur.

Python kimler tarafından kullanılıyor ?

Python şuanda birçok kuruluş tarafından kullanılmakta bunların başında; nasa, redhat, google gibi büyük kuruluşlarda var.

Kolaydır!

Python tahmin edemeyeceğiniz kadar kolay ve pratiktir.

örneğin şu kodu bir inceleyelim;

```
a = "pardus".upper()
```

gördüğünüz gibi python olağanüstü derecede nesnel bir dil burada a diye bir değişken oluşturuyoruz ve ona pardus yazısını atıyoruz ve hemen ardından upper() fonksiyonu ile bütün harflerini büyük yapıyoruz.

Neden Python ?

c++ / java gibi diller varken "neden python ?" diyebilirsiniz.

hemen sorunuza cevap verelim; python şuanda en kolay ve en güzel dil diyebiliriz çünkü yukarıda verdiğimiz örneği yapabilen bir dil şuanda yok yani python bir benzersiz!

İşletim Sistemimde Python ?

Python c ile yazılmıştır ve platformdan bağımsızdır.

Nerden Edinebilirim ?

Python'u www.python.org adresinden işletim sisteminize uygun olan versiyonunu indirebilirsiniz.

Bu kadar yeterli sanırım buraya kadar iştahınız bayağı kabarmıştır hemen python a başlayalım o zaman.

yazar : Oğuzhan Eroğlu (oguzhan@oguzhaneroglu.com)

- İÇİNDEKİLER -

1 - Python' a giriş

2 - Python Değişkenleri

a - Değişkenlere giriş

b - Cümle ve Sayı Değişkenleri

a - Tüpler

b - Listeler

c - Sözlükler

3 - Cümle İşlemleri

a - Kaçış Karakterleri

b - Cümle Biçimleme

4 - Denetleme İşlemleri

a - if değimi

b - while değimi

c - for değimi

5 - Fonksiyonlar

a - Fonksiyonlara giriş

b - Fonksiyon işlemleri

6 - Sınıflar

a - Sınıflara giriş

b - Sınıf işlemleri

7 - Hatalar

8 - Modul Kullanımı

a - Modul çağırımı

9 - Çok Kullanılan Moduller

a - os

b - sys

c - random

1. Bölüm - Python'a Giriş

Bu bölümde python un temel özelliklerinden bahsedeceğiz.
artık python'un bilgisayarınızda kurulu olduğunu varsayarak işe başlıyoruz.

python yorumlayıcısını açıyoruz.
ve karşımıza aşağıdaki gibi bir görüntü geliyor.

```
Python 2.4.4 (#1, May 14 2007, 16:54:03)
```

```
[GCC 3.4.6] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

Hemen buradan bahsedelim bu etkileşimli python yorumlayıcısıdır.

```
>>> bir kod girilmesinin beklendiği anlamına gelir  
... 1. koddan sonraki kodların beklendiğini ifade eder.
```

şimdi bi işlem yapalım

```
>>> 2 + 5
```

```
7
```

```
>>>
```

yorumlayıcıya "2 + 5" yazdık ve sonucunu verdi işte python da bu şekilde işlem yapabiliriz

Kısaca matematiksel işlemlerin karakterlerinden bahsedelim

+ , toplama yapar

- , çıkarma işlemi yapar

/ , bölme işlemi yapar

* , karakteri ise çarpma işlemi yapar.

şimdi işlemleri daha değişik yollardan yapalım.

```
>>> (2 + 2) - (3 + 2)
```

```
-1
```

```
>>>
```

gördüğünüz gibi işlemleri ayırabiliyoruz.

birde print fonksiyonuna bakalım,

```
>>> print "Merhaba Python'lu Dünya!"
```

```
Merhaba Python'lu Dünya!
```

```
>>>
```

burada gördüğünüz gibi print fonksiyonu ekranda bir yazıyı göstermek için kullanılıyor.

raw_input Fonksiyonu

Kullanıcıdan bilgi almak için kullanılır.
kullanımı;

```
>>> a = raw_input("yazi girin: ")
```

```
yazi girin: deneme
```

```
>>> print a
```

```
deneme
```

```
>>>
```

şeklindedir

Açıklama Satırları

Her programcı uygulamasını yazarken bazı yerlere açıklamalar eklemeye ihtiyaç duyar python da bu olayı # karakteri ile yapabilirsiniz.

örneğin;

```
>>> #Deneme
```

```
...
```

```
>>>
```

gördüğünüz gibi başında # olan satırlar yorumlayıcı tarafından okunmamakta.

2 - Bölüm - Python Değişkenleri

Bu bölümde python değişkenlerinden bahsedeceğiz.

Hemen başlayalım,

```
>>> a = "pardus"
```

```
>>>
```

şeklinde bir kod yazdık peki ne oldu burda ?

hemen açıklayalım, burada a adında bir değişken oluşturduk ve ona pardus cümlesini yani "pardus" u atadık

devam edecek olursak;

```
>>> print a
```

```
pardus
```

```
>>>
```

burada a değişkenini print yaptık yani ekrana yazdırdık.

şöylede yapabiliriz.

```
>>> print "Özgürlük İçin...", a
```

```
Özgürlük İçin... pardus
```

```
>>>
```

bu konuları ileride ayrıntılı olarak göreceğiz bu nedenle şimdi üzerinde fazla durmuyoruz.

eğer bir değişkene bir sayı atayacaksa onu " " karakterleri olmadan yazmalıyız yani bir örnek verecek olursak

```
>>> b = "2"
```

```
>>> c = "3"
```

```
>>> print b + c
```

```
>>>
```

burada neden böyle oldu? dediğinizi duyar gibiyim hemen açıklayalım

" ların içine yazılan değerler string dir yani yazı bunları + ifadesi kullanarak toplarsak birleştirmiş oluruz bir örnek daha verecek olursak;

```
>>> deger1 = "gnu"
```

```
>>> deger2 = "linux"
```

```
>>> print deger1 + deger2
```

```
gnulinux
```

```
>>>
```

bu şekilde daha açıklayıcı oldu sanırım :)

bir de şöyle yapalım,

```
>>> f = 5
```

```
>>> e = 2
```

```
>>> print f + e
```

```
7
```

```
>>>
```

şimdi elimizde b ve c değişkenleri hala mevcut,

bunları şöyle yaparsak

```
>>> b = int(b)
```

```
>>> c = int(c)
```

```
>>> print b + c
```

```
5
```

```
>>>
```

peki burada ne oldu ? hemen açıklayalım

elimizdeki b ve c adında 2 değişken vardı bunların değerleri b = "2" c = "3" dü bunları toplamıyordu birleştiriyordu ama şimdi topladı

işte burada **int** fonksiyonunu kullandık peki nasıl olduda kullandık ?

b = dedik yine b ye atıyoruz değeri int(b) dedik yani b değişkenini int yani sayı yap dedik ve tekrar b ye atadık

şimdi ben ötekini yaparım dediğinizi duyar gibiyim

```
>>> b = str(b)
```

```
>>> c = str(c)
```

```
>>> print b + c
```

```
23
```

```
>>>
```

evet burayı anladınız sanırım hemen açıklayalım;

bu sefer int i cümleye yani str ye çevirmek için **str** fonksiyonunu kullandık ve tekrar + ile topladık sonuç 23 çıktı yani b ve c tekrar cümle oldu.

Tüpler

Tüpler çoklu eleman için kullanılır ve içeriğini değiştiremeyiz hemen bir örnek

verelim;

```
>>> a = ("pardus", "debian", "ubuntu")
```

```
>>> a
```

```
('pardus', 'debian', 'ubuntu')
```

```
>>>
```

şeklinde kullanılırlar

Listeler

Listeler tıpkı tüpler gibidir fakat içeriğini değiştirebiliriz.

```
>>> b = ["pardus", "debian", "ubuntu"]
```

```
>>> b
```

```
['pardus', 'debian', 'ubuntu']
```

```
>>>
```

şeklinde kullanılırlar.

Append Fonksiyonu

Bir listeye bir eleman eklemek için kullanılır.

```
>>> b.append("kubuntu")
```

```
>>> b
```

```
['pardus', 'debian', 'ubuntu', 'kubuntu']
```

```
>>>
```

şeklinde kullanılır.

Remove Fonksiyonu

Bir listeden eleman silmek için kullanılır.

kullanımı,

```
>>> b.remove("ubuntu")
```

```
>>> b
```

```
['pardus', 'debian', 'kubuntu']
```

```
>>>
```

şeklindedir.

Sözlükler

Sözlükler her programcının kullanmak zorunda olduğu özelliklerdendir.

ve {} karakterleri arasına yazılırlar, hemen bir örnek verelim;

```
>>> c = {"pardus": "ozgurluk", "windows": "kisiltama"}
```

```
>>> c
```

```
{'windows': 'kisiltama', 'pardus': 'ozgurluk'}
```

```
>>>
```

şeklinde kullanılırlar.

keys Fonksiyonun

```
>>> c.keys()
```

```
['windows', 'pardus']
```

```
>>>
```

şeklinde kullanılır.

has_key Fonksiyonu

Bir sözlükte bir elemanın olup olmadığını kontrol etmek için kullanılır.

kullanımı;

```
>>> c.has_key("pardus")
```

```
True
```

```
>>> c.has_key("ubuntu")
```

```
False
```

```
>>>
```

şeklindedir.

Son olarak bir örnek verip sözlükleri bitirelim.

```
>>> d = {"pardus", "debian", "ubuntu"}: "ozgurluk"}
```

```
>>> d
```

```
{('pardus', 'debian', 'ubuntu'): 'ozgurluk'}
```

```
>>>
```

type Fonksiyonu

bir değişkenin cinsini öğrenmek için kullanılır.

```
>>> d = {"pardus", "debian", "ubuntu"}: "ozgurluk"}
```

```
>>> type(d)
```

```
<type 'dict'>
```

```
>>>
```

```
>>> a = "Pardus"
```

```
>>> type(a)
```

```
<type 'str'>
```

```
>>>
```

şeklinde kullanılır.

3. Bölüm Desen İşlemleri

Bu bölümde desen işlemlerinden bahsedeceğiz.

Python'da desen işlemleri oldukça kolaydır, hemen bir giriş yapalım.

```
>>> yazi = "Ozgurluk icin Pardus"
```

```
>>> yazi[0]
```

```
'O'
```

```
>>>
```

burada şöyle Bir şey yaptık yazi adında bir değişken tanımladık ve onun 1. karakterini aldık bunu nasıl yaparız?

degiskenadi[elemannumarasi] şeklinde...

peki ama burda [] içine 0 yazdık? işte burda kafanız karışmasın bir değişkenin 1. elemanı alınırken [] ların içine her zaman 0 yazılır.

birde sondan 1. elemanı alalım;

```
>>> yazi[-1]
```

```
's'
```

```
>>>
```

bu sefer son eleman için -1 yazdık yani sondan başlamak için - kullanırız ama -0 olamayacağı için sondan 1. eleman -1 ile alınıyor.

Kaçış Karakterleri

Şimdi bağzı desen işleme özelliklerini göreceğiz

```
>>> print "Özgürlük için...\nPardus"
```

```
Özgürlük için...
```

```
Pardus
```

```
>>>
```

gördüğünüz gibi 2 satır da verdi çıktıyı peki neden böyle oldu? çünkü \n kaçık karakterini kullandık

birde şöyle Bir şey yapalım.

```
>>> print "Özgürlük için...\n\tPardus"
```

```
Özgürlük için...
```

```
    Pardus
```

```
>>>
```

bu seferde alt satırda olacak olan yazı içeriye doğru kaydı bunuda \t kaçış karakteri ile yapıyoruz.

Desen Biçimleme

Her programcı ister istemez desen biçimlemeye ihtiyaç duyacaktır.

hemen bir örnek verelim.

```
>>> print "%s %s" % ("Özgürlük İçin", "Pardus")
```

```
Özgürlük İçin Pardus
```

```
>>>
```

gibi kullanılır "" ların arasına istediğinizi yazabilirsiniz örnek.

```
>>> print "%s...%s" % ("Özgürlük İçin", "Pardus")
```

```
Özgürlük İçin...Pardus
```

```
>>>
```

Upper Fonksiyonu

bir cümlenin bütün harflerini büyük yapmak için kullanılır.

```
>>> "pardus".upper()
```

```
'PARDUS'
```

```
>>>
```

Lower Fonksiyonu

bir cümlenin bütün harflerini küçük yapmak için kullanılır.

```
>>> "PARDUS".lower()
```

```
'pardus'
```

>>>

Capitalize Fonksiyonu

Bir cümlenin sadece baş harfini büyük yapmak için kullanılır.

```
>>> "pardus".capitalize()
```

```
'Pardus'
```

>>>

4. Bölüm Denetleme İşlemleri

Buraya kadar yazdığımız uygulamalarda programın akışını kullanıcı belirleyemiyordu artık buraya kadar geldiğinize göre python u sevdiniz demektir hemen bu bölümümüzü anlatalım.

İf Değimi

Şimdi en sevdiğiniz metin editörünü açıp deneme.py adında bir dosya kaydedin. ve ilk çalışır programımızı hemen yazalım :)

```
print \
```

```
"""
```

```
1 - Annemi
```

```
2 - Babamı
```

```
3 - İkiside Aynı
```

```
"""
```

```
a = raw_input("söyleyin ?: ")
```

```
if a == "1":
```

```
    print "tamam anneni daha çok seviyorsun."
```

```
if a == "2":
```

```
    print "tamam babamı daha çok seviyorsun."
```

```
if a == "3":
```

```
    print "aferin, ikisinide aynı seviyorsun :)"
```

çalıştırmak için konsolda python deneme.py komutunu vermeniz yeterli.

Şimdi if değimini açıklayalım.

burada a değerine kullanıcının girdiği veriyi atıyoruz:

ve kullanıcı 1 2 3 değerlerinden birini girince istediğimiz işlemi yaptırıyoruz.

if(eğer) a(değişkeninin değeri) ==(eşitse) 3 e alttaki kodları işlet şeklinde bir açıklama yapabiliriz bu kodlar için.

birde burda : karakteri var buda bir alt girintideki kodların işletilmesi için kullanılır.

eğer girintileri kullanmazsanız program çalışmaz python da başlangıç ve bitiş değimleri yerine girintiler(bloklar) kullanılır.

Peki ya eşit değilse demek için ne yapacağız?

eğer a eşit değilse "1" e şeklinde bir ifade yazarsanız == yerine != kullanmanız gerekiyor.

aynı şekilde if $a > 3$; , if $a < 3$: şeklinde ifadeler de kullanabilirdik

peki ya büyük veya eşit demek için ne yapacağız ?

if $a \geq 3$: şeklinde bir ifade kullanılır.

en son birde \ ve "" karakterlerini anlatalım \ kodların okunmasını bir alttaki satıra bırakır "" karakterini ise " karakteri gibi düşünebilirsiniz tek farkı "" lar içerisinde istediğiniz kadar satır ile yazı yazabilmenizdir.

while Değimi

while döngü yazmak için kullanılır.

kullanımı;

while karşılaştırma:

```
işletilecek_kodlar
```

şeklindedir.

örnek bir program yazacak olursak.

```
a = 3
```

```
while a != 0:
```

```
    a = a - 1
```

```
    print "a'nın şimdiki değeri, %s" % (a)
```

for Değimi

for bir çeşit döngüdür

```
>>> for a in "abc":
```

```
...     print a
```

```
...
```

```
a
```

```
b
```

```
c
```

```
>>>
```

şeklinde kullanılır.

break Değimi

bir döngüyü bitirmek için kullanılır.

```
a = ""
```

```
while a != "abc":
```

```
    print "..."
```

```
    b = raw_input("b nin değerini girin: ")
```

```
    if b == "pardus":
```

```
        break
```

5. Bölüm Fonksiyonlar

Artık moduler programlamaya tamamen geçiyoruz! şuana kadar yazdığımız

programlarda içerik moduler değil şimdiki fonksiyonları öğrenelim.

fonksiyonlar def hazır fonksiyonu ile tanımlanır.

kullanımı;

```
def fonksiyon_adi():
```

```
    isletilecek_kodlar
```

şeklindedir.

örnek bir uygulama yazalım.

```
def a():
```

```
    print "..."
```

```
a()
```

şeklinde bir uygulama yazabiliriz

fonksiyonlar çağırılırken fonksiyonadi() şeklinde çağırılır.

Fonksiyon İşlemleri

fonksiyonlara parametre vermek için () karakterlerinin arasına girilecek değerleri yazarız.

```
def a(b, c):
```

```
    print b + c
```

```
a(2, 3)
```

şeklinde...

global Fonksiyonu

bir fonksiyon kendisinin dışında kalan kodların içerisindeki değişkenleri görmez bunun için global hazır fonksiyonu kullanılır.

ayrıca fonksiyonlar birbirlerindeki değişkenlere erişemezler.

```
d = "pardus"
```

```
def a(b, c):
```

```
    global d
```

```
    print d
```

```
    print b + c
```

```
a(2, 3)
```

şeklinde kullanılır.

5. Bölüm Sınıflar

fonksiyonların başka fonksiyonlardaki değişkenlere erişemediğini öğrenmiştik.

işte burada sınıflar devreye giriyor, sınıfların değişkenlerine her yerden erişilebilir.

sınıfların kullanımı;

```
class sınıf_adi:
```

```
    def __init__(self):
```

```
        sınıfın değişkenleri burada tanımlanır.
```

```
def fonksiyon_adi(self):
```

```
    isletilecek_kod
```

şeklindedir

bir sınıfa ait bütün değişkenler self e bağlanmalıdır

yani `__init__` fonksiyonunda tanımlayacağımız bütün değişkenler `self.degiskenadi` şeklinde tanımlanır.

örnek bir uygulama yazalım.

```
class a:
```

```
    def __init__(self):
```

```
        self.deger1 = "pardus"
```

```
        self.deger2 = "linux"
```

```
    def b(self):
```

```
        print self.deger1, self.deger2
```

```
S = a()
```

```
print S.deger1
```

```
S.b()
```

7. Bölüm Hatalar

Her programcı hatalarla baş etmek zorundadır, python bu çok kolaydır.

örnek bi program yazalım ctrl c yapınca KeyboardInterrupt hatası gerçekleşiyor ve bu durumda programdan çıkılıyor hemen bu duruma el koyalım.

```
try:
```

```
    a = raw_input("..")
```

```
except(KeyboardInterrupt):
```

```
    print "hata oluştu..."
```

```
rohanrhu@pardus python $ python deneme.py
```

```
..hata oluştu...
```

```
rohanrhu@pardus python $
```

şeklinde bir sonuç alıyoruz

peki program sonlanmasın dersiniz yani hatayı pas geçelim istiyorsak ne yapacağız? bunu merak ediyorsanız pass fonksiyonunu okuyun.

Pass Fonksiyonu

```
try:
```

```
    a = int(raw_input(".."))
```

```
except:
```

```
    pass
```

```
b = raw_input("...")
```

şeklinde kullanılır.

8. Bölüm Modul Kullanımı

python da modulleri `import moduladi` şeklinde çağırabilirsiniz.

eğer bir modulu çağırıp başka bi isimle kullanmak isterseniz.

import modülüne *as* istenilen isim şeklinde çağırabilirsiniz.

eğer bir modül içerisinde belirli fonksiyonları çağıracaksanız *from modülüne import istenilenfonksiyon* veya bütün fonksiyonları çağıracaksanız *from modülüne import ** şeklinde kullanabilirsiniz.

9. Bölüm Çok Kullanılan Modüller

Os Modülü

os modülü işletim sistemi ile ilgili işlemleri yapan modüldür.

os modülünün en çok kullanılan fonksiyonlarını şimdi göreceğiz.

getcwd()

hangi dizinde çalışıldığını öğrenir.

```
>>> os.getcwd()
```

```
'/home/rohanrhu'
```

```
>>>
```

chdir()

Dizine girmek için kullanılır.

```
>>> os.chdir("/home/rohanrhu/python")
```

```
>>> os.getcwd()
```

```
'/home/rohanrhu/python'
```

```
>>>
```

system()

işletim sisteminin konsol işlemlerini yapmak için kullanılır.

```
>>> os.chdir("/")
```

```
>>> os.system("ls")
```

```
bin boot dev etc home lib lost+found media mnt none opt proc root sbin  
sys tmp usr var
```

```
0
```

```
>>>
```

Sys Modülü

sys modülü uygulama ile ilgili işlemleri yapar.

en çok `exit()` (uygulamayı kapatır) fonksiyonu kullanılır.

```
>>> import sys
```

```
>>> sys.exit()
```

```
rohanrhu@pardus ~ $
```

Random Modülü

random modülü rastgele seçimler yapmak için kullanılır.

randrange()

rastgele sayı seçmek için kullanılır

```
>>> random.randrange(99)
```

```
45
```

```
>>>
```

choice()

liste veya tüplerden rastgele eleman seçmek için kullanılır.

```
>>> a = ("pardus", "debian", "ubuntu")
```

```
>>> random.choice(a)
```

```
'pardus'
```

```
>>>
```

şeklinde kullanılabilir.

Dökümanımız burada bitiyor.

elimden geldiğince açık olmaya çalıştım, umarım python a yeni başlayanlara faydalı olur.