modüller

Modül:

- → diğer Python programları tarafından kullanılmak üzere Python tanımlamaları ve
- → cümlelerini içeren bir **dosya** dır.
- → Standart/yerleşik kütüphanenin parçasından modüller: doctest, string

pydoc

→ pydoc modülü, sistemde kurulu Python kütüphanelerinde arama yapmada kullan

```
$ pydoc -g
```

→ “open browser”i deneyin

→ sistemdeki Python tarafından bulunan bütün python kütüphanelerinin bir listesidir.

ör. keyword

ör. keyword

```
Functions
iskeyword = __contains__(...)
    x.__contains__(y) <==> y in x.

Data
__all__ = ['iskeyword', 'kwlist']
kwlist = ['and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except',
'exec', 'finally', 'for', 'from', 'global', 'if',
'import', 'in', 'is', ...]
```

renklendirme

Çoğu modülün belgeleri üç renkli kod kısımları içermektedir:

- → Sınıflar pembe
- → Fonksiyonlar turuncu
- → Veri yeşil

özet

- → Sınıflar daha sonraki bölümlerde anlatılacaktır,
- → ama şimdilik pydoc'u modüller içerisindeki **fonksiyonları** ve
- → **verileri** görmek için kullandığımızı bilin

keyword modülü

keyword modülü: fonksiyonları

```
>>> from keyword import *
>>> iskeyword('for')
True
>>> iskeyword('all')
False
>>>
```

keyword modülü: verileri

```
>>> from keyword import *
>>> print kwlist
['and', 'as', 'assert', 'break'
'else', 'except', 'exec', 'fina
'in', 'is', 'lambda', 'not', 'o
'while', 'with', 'yield']
>>>
```

modül oluşturma

→ betik dosyasından çok farklı bir şey değil!

```
1  #  seqtools.py
2  #
3  def remove_at(pos, seq):
4      return seq[:pos] + seq[pos+1:]
```

kendi modülünüzü kullanma

→ hem betikte hem de kabukta kullanabilirsiniz

→ bunun için içe aktarmak (_import_) gerekir

→ bu ise iki şekilde olur

ilk yol

```
>>> from seqtools import remove_at
>>> s = "A string!"
>>> remove_at(4, s)
'A sting!'
```

ikinci yol

```
>>> import seqtools
>>> s = "A string!"
>>> seqtools.remove_at(4, s)
'A sting!'
```

özet

- ilkinde remove_at daha önce gördüğümüz fonksiyonlar gibi çağrıldı.
- ikincisinde modülün ismi ve bir nokta (.) fonksiyon isminden önce yazıldı.
- iki durumda da dosyayı içe aktarırkan .py uzantısını yazmadığımıza dikkat edin.
- modül kullanımı çok büyük programları **yönetilebilir** büyüklükte parçalara bölmemize
- ve **ilişkili** parçaları birlikte tutmamıza yaramaktadır.

isim uzayı

- → isim uzayı sözdizimsel bir kaptır.
- → her modül kendi isim uzayını belirler,
- → aynı ismi farklı modüllerde tanımlama problemi oluşmaz

ilk modül

```
# module1.py

question = "What is the meaning of life,\
           the Universe, and everything?"
answer = 42
```

ikinci modül

```
# module2.py

question = "What is your quest?"
answer = "To seek the holy grail."
```

isim uzayı

her iki modülü içeri aktarıp içerisindeki (aynı isimli) değişkenlere erişebiliriz

```
>>> import module1
>>> import module2
>>> print module1.question
What is the meaning of life, the Universe, and everything ?
>>> print module2.question
What is your quest ?
>>> print module1.answer
42
>>> print module2.answer
To seek the holy grail.
>>>
```

→ Eğer from module1 import * ve from module2 import * yazsaydık???

isim uzayı

her iki modülü içeri aktarıp içerisindeki (aynı isimli) değişkenlere erişebiliriz

```
>>> import module1
>>> import module2
>>> print module1.question
What is the meaning of life, the Universe, and everything ?
>>> print module2.question
What is your quest ?
>>> print module1.answer
42
>>> print module2.answer
To seek the holy grail.
>>>
```

→ Eğer from module1 import * ve from module2 import * yazsaydık???

→ **isimlendirme çakışması**, sonuç: module1.question|answer erişilemez

isim uzayı: fonksiyonlar

→ fonksiyonlar da kendi isim uzaylarına sahiptir

```
1  def f():
2      n = 7
3      print "printing n inside of f: %d"  % n
4
5  def g():
6      n = 42
7      print "printing n inside of g: %d"  % n
8
9  n = 11
10 print "printing n before calling f: %d"  % n
11 f()
12 print "printing n after calling f: %d"  % n
13 g()
14 print "printing n after calling g: %d"  % n
```

→ çıktısı

```
1  printing n before calling f: 11
2  printing n inside of f: 7
3  printing n after calling f: 11
4  printing n inside of g: 42
5  printing n after calling g: 11
```

→ üç n burada çakışmaz çünkü her biri ayrı isim uzayındadır.

isim uzayı: özet

İsim uzayları birden fazla programcının **aynı** projede *isim çakışmalarıyla* karşılaşmadan **birlikte** çalışmasına olanak sağlar.

özellikler ve nokta işleci

→ modül içerisinde tanımlanmış olan değişkenlere modülün **özellikleri** denir.

→ Bu özelliklere nokta işleci (.) ile **erişilir**.

→ ör. module1 ve module2'nin question özelliklerine

→ module1.question ve module2.question şeklinde erişilmektedir.

özellikler ve nokta işleci

modüle ait fonksiyonlara erişmek için de nokta işleci kullanılır

```
>>> import string
>>> string.capitalize('maryland')
'Maryland'
>>> string.capwords("what's all this, then, amen?")
"What's All This, Then, Amen?"
>>> string.center('How to Center Text Using Python', 70)
'                    How to Center Text Using Python                    '
>>> string.upper('angola')
'ANGOLA'
>>>
```

karakter dizisi ve liste metotları

→ string modülündeki bir çok işlev karakter dizisi nesnesine de eklenmiştir

→ yeni eklentiler **metod** olarak adlandırılır

```
>>> 'maryland'.capitalize()
'Maryland'
>>> "what's all this, then, amen?".title()
"What'S All This, Then, Amen?"
>>> 'How to Center Text Using Python'.center(70)
'                    How to Center Text Using Python
>>> 'angola'.upper()
'ANGOLA'
>>>
```

liste metotları

nokta işleci nesnelerin yerleşik metotlarına erişmek için kullanılır

```
>>> mylist = []
>>> mylist.append(5)
>>> mylist.append(27)
>>> mylist.append(3)
>>> mylist.append(12)
>>> mylist
[5, 27, 3, 12]
>>>
```

liste metotları

diğerleri

Method Name	Use	Explanation
append	`alist.append(item)`	Adds a new item to the end of a list
insert	`alist.insert(i,item)`	Inserts an item at the ith position in a list
pop	`alist.pop()`	Removes and returns the last item in a list
pop	`alist.pop(i)`	Removes and returns the ith item in a list
sort	`alist.sort()`	Modifies a list to be sorted
reverse	`alist.reverse()`	Modifies a list to be in reverse order
del	`del alist[i]`	Deletes the item in the ith position
index	`alist.index(item)`	Returns the index of the first occurrence of `item`
count	`alist.count(item)`	Returns the number of occurrences of `item`
remove	`alist.remove(item)`	Removes the first occurrence of `item`

Table 1.2: Methods Provided by Lists in Python

kullanım

kullanım

```
>>> mylist.insert(1, 12)
>>> mylist
[5, 12, 27, 3, 12]
>>> mylist.count(12)
2
>>> mylist.extend([5, 9, 5, 11])
>>> mylist
[5, 12, 27, 3, 12, 5, 9, 5, 11])
>>> mylist.index(9)
6
>>> mylist.count(5)
3
>>> mylist.reverse()
>>> mylist
[11, 5, 9, 5, 12, 3, 27, 12, 5]
>>> mylist.sort()
>>> mylist
[3, 5, 5, 5, 9, 11, 12, 12, 27]
>>> mylist.remove(12)
>>> mylist
[3, 5, 5, 5, 9, 11, 12, 27]
>>>
```

metin dosyalarını okuma ve yazma

→ programın çalışırken kullandığı bellek-RAM, geçici

→ kalıcı olarak saklamak istersek dosya kullanın

→ dosyayı kullanmak için bir isim verip, açmak gerekir

```
1  >>> myfile = open('test.dat', 'w')
2  >>> print myfile
3  <open file 'test.dat', mode 'w' at 0x2aaaaab80cd8>
```

→ open fonksiyonu iki argüman almaktadır.

1. dosyanın ismi
2. mod: 'w' modu= yazma

dosyaya yazma

dosyaya yazma veya veri koyma

```
>>> myfile.write("Now is the time")
>>> myfile.write("to close the file")
```

→ dosyaya yazma işimizin bittiğini söylememiz gerekir

→ bunun için kapama yapılır

```
>>> myfile.close()
```

dosyadan veri okuma

dosyadan veri okuma

```
>>> myfile = open('test.dat', 'r')
```

dosya yoksa

```
>>> myfile = open('test.cat', 'r')
IOError: [Errno 2] No such file or directory: 'test.cat'
```

dosyadan veri okuma

read işlevi argümansız işletilirse tüm dosyayı tek seferde okur

```
>>> text = myfile.read()
>>> print text
Now is the timeto close the file
```

→ okunacak karakter sayısını girebiliriz

```
>>> myfile = open('test.dat', 'r')
>>> print myfile.read(5)
Now i
```

→ okumaya devam edebiliriz

```
>>> print myfile.read(1000006)
s the timeto close the file
>>> print myfile.read()

>>>
```

→ dosyada veri kalmayınca boş dizgi döner

dosya kopyala

dosya kopyala

```
1   def copy_file(oldfile, newfile):
2       infile = open(oldfile, 'r')
3       outfile = open(newfile, 'w')
4       while True:
5           text = infile.read(50)
6           if text == "":
7               break
8           outfile.write(text)
9       infile.close()
10      outfile.close()
11      return
```

→ şifrele sakla: 50 karakterin yerini değiştir (kontrollü shuffle)

→ şifrele sakla: her bir harfi başka bir harfle değiştir

→ hash'ini sakla

→ filtrele sakla: küçük harfleri filtrele

metin dosyaları

metin dosyası

→ yazdırılabilir karakterler ve beyaz boşluklar içeren,

→ satırlar şeklinde düzenlenmiş,

→ satırları yeni satır karakterleriyle ayrılmış

dosyadır.

dosyaya yazma

dosyaya yazma

```
>>> outfile = open("test.dat","w")
>>> outfile.write("line one\nline two\nline three\n")
>>> outfile.close()
```

→ yeni satır karakterine dikkat

şimdi okuyalım

```
>>> infile = open("test.dat","r")
>>> print infile.readline()
line one

>>>
```

satırları okuma

→ tüm satırları okumak

```
>>> print infile.readlines()
['line two\012', 'line three\012']
```

→ dosya sonuna ulaşınca readline boş satır, readlines boş liste döndürür

```
>>> print infile.readline()

>>> print infile.readlines()
[]
```

örnek

satır işleme programı

```
1   def filter(oldfile, newfile):
2       infile = open(oldfile, 'r')
3       outfile = open(newfile, 'w')
4       while True:
5           text = infile.readline()
6           if text == "":
7              break
8           if text[0] == '#':
9              continue
10          outfile.write(text)
11      infile.close()
12      outfile.close()
13      return
```

→ dosyayi ikiye ayirin bir dosya aciklamalari, digeri ise programi icersin

dizinler

- → dosyalar, dosya sisteminde dizinler içerisinde saklanırlar
- → dizinler ise hem dosyalara hem de alt dizinlerine ev sahipliği ederler
- → açmaya/kaydetmeye çalıştığınız dosyalar **aynı** dizinde olmalıdır
- → **farklı** dizinlerle çalışmak istersek?
- → konumunu (**path**) söylemeliyiz

```
>>> wordsfile = open('/usr/share/dict/words', 'r')
>>> wordlist = wordsfile.readlines()
>>> print wordlist[:5]
['\n', 'A\n', "A's\n", 'AOL\n', "AOL's\n", 'Aachen\n']
```

counting Letters

ord fonksiyonu bir karakterin tamsayı temsilini döndürmektedir:

```
1  >>> ord('a')
2  97
3  >>> ord('A')
4  65
5  >>>
```

→ 'Apple' < 'apple', True veya False ????

chr işlevi

- chr fonksiyonu ord fonksiyonunun tersidir.
- tamsayı --> karakter

```
>>> for i in range(65, 71):
...     print chr(i)
...
A
B
C
D
E
F
>>>
```

örnek

→ countletters.py: karakter histogramı

```
1   def display(i):
2       if i == 10: return 'LF'
3       if i == 13: return 'CR'
4       if i == 32: return 'SPACE'
5       return chr(i)
6
7   infile = open('alice_in_wonderland.txt', 'r')
8   text = infile.read()
9   infile.close()
10
11  counts = 128 * [0]
12
13  for letter in text:
14      counts[ord(letter)] += 1
15
16  outfile = open('alice_counts.dat', 'w')
17  outfile.write("%-12s%s\n" % ("Character", "Count"))
18  outfile.write("================\n")
19
20  for i in range(len(counts)):
21      if counts[i]:
22          outfile.write("%-12s%d\n" % (display(i), counts[i]))
23
24  outfile.close()
```

sys modülü ve argv

→ sys modülü python yorumlayıcının çalıştığı ortam hakkında bilgi/erişim sağlar

```
>>> import sys
>>> sys.platform
'linux2'
>>> sys.path
['', '/home/jelkner/lib/python', '/usr/lib/python25.zip', '/usr/lib/python2.5
'/usr/lib/python2.5/plat-linux2', '/usr/lib/python2.5/lib-tk',
'/usr/lib/python2.5/lib-dynload', '/usr/local/lib/python2.5/site-packages',
'/usr/lib/python2.5/site-packages', '/usr/lib/python2.5/site-packages/Numeric
'/usr/lib/python2.5/site-packages/gst-0.10',
'/var/lib/python-support/python2.5', '/usr/lib/python2.5/site-packages/gtk-2.
'/var/lib/python-support/python2.5/gtk-2.0']
>>> sys.version
'2.5.1 (r251:54863, Mar  7 2008, 04:10:12) \n[GCC 4.1.3 20070929 (prerelease)
(Ubuntu 4.1.2-16ubuntu2)]'
>>>
```

argv

- → argv değişkeni Python betiği çalıştırıldığında komut satırından okunan
- → karakter dizilerinin listesini tutmaktadır.
- → Bu komut satırı argümanları program başlarken programa bilgi geçirmeye yardımcı olur.

```
1  #
2  # demo_argv.py
3  #
4  import sys
5
6  print sys.argv
```

böyle deneyin

```
1  $ python demo_argv.py this and that 1
2  ['demo_argv.py', 'this', 'and', 'that',
3  $
```

- → dosya kopyalamayı vd bununla güncelleyin
- → ilk eleman programın ismi
- → bir de böyle deneyin (beyaz boşluklar)

```
1  $ python demo_argv.py "this and" that
2  ['demo_argv.py', 'this and', 'that', '
3  $
```

örnek

→ komut satırından girilen sayı dizisinin toplamını alan program:

```
1  #
2  # sum.py
3  #
4  from sys import argv
5
6  nums = argv[1:]
7
8  for index, value in enumerate(nums):
9      nums[index] = float(value)
10
11 print sum(nums)
```

→ from <module> import <attribute> şeklinde içe aktarmayı kullanmaktayız,

→ böylece argv modülün isim uzayına alınmış oluyor.

→ demo

```
1  $ python sum.py 3 4 5 11
2  23
3  $ python sum.py 3.5 5 11 100
4  119.5
```

dizinlerde dolaşma

cook/python/src altındaki

- → dir_file_processing.py
- → dir_walk.py
- → os_listdir.py
- → pdfnot.py

sıra sizde

myreplace: split, join kullanılacak

```
1  def myreplace(old, new, s):
2      """
3      Replace all occurances of old with new in the string s.
4
5          >>> myreplace(',', ';', 'this, that, and, some, other, thing')
6          'this; that; and; some; other; thing'
7          >>> myreplace(' ', '**', 'Words will now be separated by stars.')
8          'Words**will**now**be**separated**by**stars.'
9      """
```

diğerleri

→ **d10_wordtools.py**

→ unsorted_fruits.txt dosyası her biri farklı bir karakterle başlayan 26 tane meyve içermektedir. sırala

→ medyan değer

```
$ python median.py 3 7 11
7
$ python median.py 19 85 121
85
$ python median.py 11 15 16 22
15.5
```

→ countletters.py programını dosyayı komut satırı argümanı olarak alacak şekilde değiştirin. Çıktı dosyasını isimlendirmeyi nasıl çözersiniz?

CSV

oku

```
1  import csv
2  reader = csv.reader(open("some.csv", "rb"))
3  for row in reader:
4      print row
```

yazma

```
1  # File: csv-example-4.py
2
3  import csv
4  import sys
5
6  data = [
7      ("And Now For Something Completely Different", 1971, "Ian MacNaughton"),
8      ("Monty Python And The Holy Grail", 1975, "Terry Gilliam, Terry Jones"),
9      ("Monty Python's Life Of Brian", 1979, "Terry Jones"),
10     ("Monty Python Live At The Hollywood Bowl", 1982, "Terry Hughes"),
11     ("Monty Python's The Meaning Of Life", 1983, "Terry Jones")
12 ]
13
14 writer = csv.writer(sys.stdout)
15
16 for item in data:
17     writer.writerow(item)
18
19 $ python csv-example-4.py
20 And Now For Something Completely Different,1971,Ian MacNaughton
21 Monty Python And The Holy Grail,1975,"Terry Gilliam, Terry Jones"
22 Monty Python's Life Of Brian,1979,Terry Jones
23 Monty Python Live At The Hollywood Bowl,1982,Terry Hughes
24 Monty Python's The Meaning Of Life,1983,Terry Jones
```