Commercio Elettronico

Python

Claudio Silvestri

(Some) Data Type

- As usual:
  - bool, int, long, float
- Not so usual:
  - complex
  - set, list, tuple, dict
- The Null object is “None”

Indentation and code writing

```
# No “{}” to identify code blocks, whitespaces matter !
# if statement
if True:
    print “True ” * 3
else:
    print “False”

# while statement
i = 0
while i<10:
    print “i is ” + str(i)
    i += 1
print “End of while statement: i=%d” % i
```

Functions and Classes

```
def square(x):
    result = x**2
    return result
print square(77)

class Rectangle:
    # everything is public
    width  = 0
    height = 0
    def __init__(self, w, h):
        self.width  = w
        self.height = h
    def area(self):
        return self.width*self.height

r = Rectangle(10,5)
print r.area()
```

Lists

```
# this is a list
vals = [1,2,3,4,5]

#accessing the list
print vals[0]==1
print vals[-1]==5
print vals[2:3]==[3]
print vals[2:-1]==[3,4]

vals.append(6)
print "len=%d" % len(vals)
print "99 ? " + str(99 in vals)

# iterating through a list
for v in vals:
    print v
```

Lists

```
# this is a peculiar list
print range(6)==[0,1,2,3,4,5]

# sorting lists
a = [3,4,1,6,5,8,9]
b = sorted(a)
print a
print b
a.sort()
a.reverse()
print a

# returning multiple values
def f(x):
    return x, x*x, x*x*x
val = f(3)
print "val=" + str(val)
val, square, cube = f(3)
print "square = %d" % square
```

Advanced functions

```
vals = [1,2,3,4,5]

# not so advanced
print min(vals), max(vals)

def square(x):
  return x*x

# borrowed from functional programming
odds = filter(lambda x:x%2==1, vals)
squares = map(square, vals)
sum = reduce(lambda x,y:x+y, vals)

print odds, squares, sum
```

More on lists

```
vals = [x for x in range(10)]

print vals

vals = [x**2 for x in range(10)]

print vals

vals = [x**2 for x in range(10) if x%2==0]

print vals
```

Dictionaries

```
eng_ita = {'one':'uno', 'two':'due', 'three':'tre'}
eng_ita['four'] = 'quattro'
print eng_ita.setdefault('five','cinque')
eng_ita['others'] = ['sei',7,8,9,10]

print eng_ita.keys()

print eng_ita.values()

for k,v in eng_ita.items():
    print k + ' = ' +  str(v)

vals = [1,2,3]
newDict=dict([(v,v*10) for v in vals])
newDict['sum'] = sum(vals)
print newDict
```

Strings

```
txt = 'is there anybody out there?'
print txt[-1]
print txt[-6:]
print txt[3:12]
words = txt.split()
print words
txt += '\n no!'
print txt

# there are other funtions such as
# replace, startswith, endswith, etc. etc.
```

Regular expressions

```
# import the regular expressions package
import re
txt = 'is there anybody out there ?'
m = re.match('.*any.*', txt)
print m!=None

# match groups
m = re.match('(.*)any(.*)', txt)
print 'match'
if m!=None:
    print m.group(0)
    print m.group(1)
    print m.group(2)
print 'search'

# search is not match
m = re.search('([a-z]*) any([a-z]*)', txt)
if m!=None:
    print m.group(0)
    print m.group(1)
    print m.group(2)
```

File I/O

```
input  = open('input.txt', 'r')
output = open('output.txt', 'w')

line = input.readline()

lines = input.readlines()

for line in input:
    output.write(line)
    print >>output, line

input.close()
output.close()
```

linea singola

tutte le linee

iteratore

File I/O

```
import fileinput

for line in fileinput.input():
    process(line)
```

Legge da standard input oppure dai file indicati nella linea di comando

urllib2

```
# download a web page

import urllib2

page = urllib2.urlopen('http://www.google.it')

for line in page:

    print line
```

Putting it all together

```
#!/usr/bin/python

# imports …
import re, sys, urllib2

def doSomething(args):
    for arg in args:
        print arg
    …

if __name__ == '__main__':
    doSomething(sys.argv)
```

For anything else ...

```
import urllib2

page = urllib2.urlopen('http://docs.python.org')

for line in page:

    print line
```

Esercizi

- Convertire tutti i caratteri di un file di testo in MAIUSCOLO
- Sostituire tutte le occorrenze di una espressione regolare in un file di testo con una stringa (entrambe da command line)
- Implementare una classe che costruisce il lessico per una collezione di file (metodo aggiorna invocato più volte con parametro nome file). Mantenere anche il numero totale occorrenze per termine e totale documenti distinti contenenti ciascun termine.
- ATTENZIONE: per non dover gestire la codifica dei caratteri, utilizzare solo caratteri 0-9 a-z A-Z e punteggiatura (es: testo in inglese)

- href=“.....”
- href\s*=\s*“([^”]*)”