Python Programming Language

Ahmet Emre Aladağ
aladagemre at gmail.com
http://www.emrealadag.com

Alcatel-Lucent
01.08.2008

Outline

1. Introduction
2. Why Python?
3. Syntax
4. Data Types
5. Control Statements
6. Input/Output, Formatting
7. Functions
8. Classes
9. Modules
10. Packages
11. Standard Library
12. Some Module Examples
13. Django
14. PyQt GUI
15. Resources

Introduction

- Platform independent
- Interpreted: Shell / File (.py)
- Byte Code (.pyc)
- Version 2.7 & Python 3K
- No Filename - Class Name match requirement
- Windows IDE: ActivePython, Komodo
- Linux IDE: Eclipse+PyDev, Eric, ActivePython, Komodo
- Py2exe
- disutils
- GUI: PyQt (Qt Designer), PyGTK (Glade), WxWidgets, Tkinter, xulrunner
- Web: Django, Turbogears, Plone
- Mobile: PyS60

Why Python?

- Functional & Fully Object Oriented
- Rapid Prototyping
- Highly flexible
- Short code, giant work
- Convenient to human-mentality
- Interactive
- Easy to learn
- Extensive Standard Library and 3rd party modules
- Can be combined with C, Fortran, COBOL, .NET (IronPython), Java (Jython) to improve performance / ease of use
- Used by NASA, Google, Yahoo, Microsoft etc.
- For every platform: Desktop/Web/Mobile
- Google AppEngine

Syntax Preview

- Indentation required
- Structures start with **:**
- Comments with **#**
- No ; required
- \ for splitting line
- '''string''' for long strings (like pre tag)

- + - / *, % for mod
- x+= for increment, no x++
- Object.property
- import math
  >>> math.hypot(3,4)
  5.0

- from os.path import *
  >>> exists("/test")
  False

Data Types

- int
- float
- long
- str
- unicode
- complex numbers
- list, tuple, dictionary, set

General Data Types

```
x = 4
y = "93"
z = int(y) + x    #97
t = y*3           #939393
m = 192893482712L
name = unicode('Dağ')

complex = 2 + 4j
print complex        #(2+4j)
print complex.img # 4
print complex.real # 2
```

```
existance = True
definition = None
```

Lists

```
ages = [39, 23, 30, 22,34]
names = ["Ahmet", "Emre"]
mixed = [21, "Emre", [3,4,9], 1.75, 88, time.time()]
ages[3]
ages[2:4] # [30, 22]
ages[0:4:2] #  [39,30]
copy_ages = ages[:] # slice copy
len(names) or names.__len__() #number of elements

ages.append('test') #Stack push
ages.pop() # Stack pop
names.append("Melih") # Queue in
names.pop(0) # Queue out (Ahmet)
```

Lists

```
ages.extend([11,12,93])
ages.insert(3, 'unknown') #Inserts 'unknown' at index 3.
ages.count(30) # counts the number of 30s.
ages.sort() #Sorts in increasing order
ages.reverse()
ages.index(12) # gives the index of element 12.
ages.remove(12) # removes the first occurance of 12.
del ages[3:] or del ages[:2]

>>> [2*age for age in ages if age > 25]
[78, 68, 60]

>>> lengths = [3,4,5,6,7]
>>> [(l, l**2) for l in lengths]
[(3, 9), (4, 16), (5, 25), (6, 36), (7, 49)]
```

Tuples

Tuples are Immutable objects
(can't be changed, like constants)

```
mytuple = (1,2,3,4,'Emre')
mytuple = mytuple, ('new','elements')
# (1,2,3,4,'Emre', ('new','elements'))

modules = (math, re, urllib)

PI, speed_limit = 3.14, 90
x,y = y,x        #Swapping
mylist = tuple([1,2,3])
mytuple = tuple(mylist)
```

```
single = 'alone',
# single = ('alone',)
x,y,z,t,p = mytuple
```

Sets

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange',
'banana']
>>> fruits = set(basket)
>>> print apple in basket
True
>>> print orange not in basket
False

sorted(set(str(91294872712032)))
['0', '1', '2', '3', '4', '7', '8', '9']
```

Sets

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                  # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b                              # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b                              # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b                              # letters in both a and b
set(['a', 'c'])
>>> a ^ b                              # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Dictionaries

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
```

Dictionaries

```
>>> tel.keys()
['guido', 'irv', 'jack']
>>> tel.has_key('guido')
True
>>> 'guido' in tel
True

>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'jack': 4098, 'guido': 4127}
>>> dict([(x, x**2) for x in (2, 4, 6)])     # use a list
comprehension
{2: 4, 4: 16, 6: 36}
```

Control Statements

- if
- range
- for, break/continue
- while, pass
- lambda, map, filter
- NO switch-case statement!

Comparison

== Equals
!= Not Equals
< <= > >= G/L Than (or Equal to)
is Same object?
in Check existance
not Negation
and or Logical

2 in [1,2,3] -> True
3 not in [1,2,3] -> False
a = b = 3
a is b -> True

if

```
if dice1 == 6 and dice2 == 6:
  print "You won!"
elif dice1 == 6 or dice2 == 6:
  print "You could win!"
else:
  print "You lost!"

if name == "Emre" or age in range(20,30): print "Done"

if a < b == c :
  print "Well done!"
```

range

range(10) -> [0,1,2,3,4,5,6,7,8,9]
range(3,5) -> [3,4]
range(1,10,3) -> [1,4,7]
range(-10, -100, -30) -> [-10, -40, -70]

Instead of switch-case

```
def one():pass; def two():pass; def three():pass
cases = {    1:one,
             2:two,
             3:three
             }
c = input("Choice:")
cases[c]()

eval("...")
```

for

```
people = ["Emre", "Melih","Mert"]
for person in people:
  print person
```

Emre
Melih
Mert

```
numbers = [92,19,37,45,73,14,75,35]
for number in numbers:
    if number > 40:
        numbers.remove(number)
```

[19, 37, **73**, 14, 35]

for - Safe modification

Safe Modification

```
numbers = [92,19,37,45,73,14,75,35]
for number in numbers[:]:
    if number > 40:
        numbers.remove(number)
```

[19, 37, 14, 35]

for - else

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print n, 'equals', x, '*', n/x
            break
    else:
        # no factor found
        print n, 'is a prime number'
```

```
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

for extras

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.iteritems():
...     print k, v
...
gallahad the pure
robin the brave
```

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print i, v
0 tic
1 tac
2 toe
```

for extras

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...         print 'What is your %s?  It is %s.' % (q, a)
...
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

while

```
while True:
  pass
#--------------------
total = 0; x=raw_input("Enter:")
while x:
  total += int(x);
  x = raw_input("Enter:")
print total
#--------------------
sum([int(raw_input()) for i in range
(1,4)] )
```

Waiting for interrupt from the keyboard

Enters input until empty string entered. Then prints the sum.

Sums 3 numbers.

lambda

Short (use-once) function definitions

```
>>> map(lambda(x): x*x*x, range(1, 11))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

>>>reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])
15 # ((((1+2)+3)+4)+5)

>>> filter(lambda(x):x % 2 != 0 and x % 3 != 0 , range(2, 25))
[5, 7, 11, 13, 17, 19, 23]
```

Input/Output

```
name = raw_input("Enter your name:")
age = input("Enter your age:")
print name, ", you are", age, "years old."
```

John , you are 30 years old.

```
f = open("file1.txt")
f.read()
f.close()
```

```
f = open("file2.txt","r")
line = f.readline(); print line
while line:
  line = f.readline(); print line
f.close()
```

```
f = open("file3.txt","w")
f.write("Test")
f.close()
```

Shelve

```
>>> import shelve
>>> profile = shelve.open("settings")
>>> profile["username"] = "emre"
>>> profile["password"] = "test"
>>> profile.close()
>>>
>>> profile = shelve.open("settings")
>>> profile["username"]
emre
```

Formatting

```
print "Name: %s" % name
print "Hello %s, you have got %d mails" % (name, mail)
Hello John , you have got 3 mails"
>>> print "Hello %(name)s, you are %(age)d years old" %
{'name': 'Emre', 'age':21}
Hello Emre, you are 21 years old

>>> 'x' + 'y'
'xy'
>>> [1,2] + [3,4]
[1, 2, 3, 4]
```

Functions

```
def square(x):
  return x**2

def add(x,y):
  return x+y

def identify(name, admin=0):
  if admin:
    print "Welcome Mr. %s" % name
  else:
    print "Welcome visitor!"

def fprintf(file, format, *args):
    file.write(format % args)
```

Functions - Calling/Aliasing

```
def fib(n):
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b
>>> fib(2000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
>>> fib
<function fib at 10042ed0>
>>> f = fib
>>> f(100)
1 1 2 3 5 8 13 21 34 55 89
```

Functions - Static variables

```
def f(a, L=[ ]):
    L.append(a)
    return L
print f(1)
print f(2)
print f(3)
```

```
[1]
[1, 2]
[1, 2, 3]
```

```
>>> def make_incrementor(n):
...     return lambda x: x + n
...
>>> f = make_incrementor(42)
>>> f(0)
42
>>> f(1)
43
```

```
def f(a, L=None):
    if L is None:
        L = [ ]
    L.append(a)
    return L
print ...
...
```

```
[1]
[2]
[3]
```

Functions - Nested

```
def viking_chorus(myfunc):
    def inner_func(*args, **kwargs):
        for i in range(8):
            myfunc(*args, **kwargs)
    return inner_func

@viking_chorus
def menu_item():
    print "Hora"
```

Documentation Strings

```
>>> def my_function():
...     """Do nothing, but document it.
...
...     No, really, it doesn't do anything.
...     """
...     pass
...
>>> print my_function.__doc__
Do nothing, but document it.

    No, really, it doesn't do anything.
```

Classes

```
class Human:
    "A simple example class"
    century = 21
    def __init__(self, name, parents = [ ]):
        self.name = name
        self.parents = parents
        print self.welcome()

    def welcome(self):
        if self.parents:
            return 'Welcome to century %d %s, the child of %s and
%s!' % (self.century, self.name, self.parents[0], self.parents[1])

        return 'Welcome to century %d %s!' % (self.century, self.
name)
```

Classes - Instantiation

```
>>> x = Human("Quiz",["Midterm","Final"])
Welcome to century 21 Quiz, the child of Midterm and Final!

>>> x.welcome
<bound method Human.welcome of <Human instance at
0xb7bdf7cc>>

>>> x.welcome()
Welcome to century 21 Quiz, the child of Midterm and Final!

>>> x.century = 39
>>> x
<Human instance at 0xb7bdf7cc>
>>> x.century
39
```

Classes - Modification

```
class MyMath:
    PI = 3.14
    def area(self, x):
        return self.PI * x * x

>>> MyMath.PI
3.1400000000000001

>>> def test(self, x):
...   return x*x

>>> MyMath.area = test
>>> x = MyMath()
>>> x.area(4)
16
```

Classes - Injection

```
>>> class Victim:
...   pass
>>> def hi(self):
...   print "Hi!"
>>> PI = 3.14
>>> Victim.hi = hi
>>> delattr(Victim, 'hi')
>>> Victim.hello = hi
>>> Victim.PI = PI
>>> Victim.
Victim.PI        Victim.__doc__     Victim.__module__  Victim.
hello

>>>Victim.PI
3.1400000000000001
```

Classes - Injection

```
class Employee:
    pass

john = Employee() # Create an empty employee record

# Fill the fields of the record
john.name = 'John Doe'
john.dept = 'computer lab'
john.salary = 1000
```

Classes - Possessive methods

```
>>> def method():
...         t = 39
...         print t
>>> def emre(): print 'TEST'

>>> method.author='emre'
>>> method.t = 11
>>> print method.author, method.t
emre 11
>>> method()
39
>>> method.fun = emre
>>> method.fun()
TEST
```

Classes - Private Variables

```
>>> class PriTest:
...         __privatevar = 23
...   def __privateMethod(self):
...         print "helo", self.__privatevar
...
>>> x = PriTest()
>>> x._
x._PriTest__privateMethod  x.__class__                x.
__module__
x._PriTest__privatevar     x.__doc__

>>> x._PriTest__privateMethod()
hello 23
```

Classes - Some internal methods

__new__(cls[, ...])
__init__(self[, ...])
__del__(self)
__repr__(self)
__str__(self)
__unicode__(self)
__lt__(self, other)
__le__(self, other)
__eq__(self, other)
__ne__(self, other)
__gt__(self, other)
__ge__(self, other)
__cmp__(self, other)
__hash__(self)
__nonzero__(self)

Classes - Inheritance

```
class Animal:
    def __init__(self, name):
        self.name = name

class Dog(Animal):
    def __init__(self, name):
        super(name)

    def wohwoh(self):
            print "Woh woh"
```

Exception Handling

```
import sys
try:
    f = open('myfile.txt')
    s = f.readline()
    i = int(s.strip())
except IOError, (errno, strerror):
    print "I/O error(%s): %s" % (errno, strerror)
except ValueError:
    print "Could not convert data to an integer."
except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
else:    print "Finished reading."
finally:
    f.close()
```

Exception Handling

```
try:
   raise Exception('spam', 'eggs')
except Exception, inst:
   print type(inst)     # the exception instance
   print inst.args      # arguments stored in .args
   print inst           # __str__ allows args to printed directly
   x, y = inst          # __getitem__ allows args to be unpacked directly
   print 'x =', x
   print 'y =', y
#output:
<type 'exceptions.Exception'>
('spam', 'eggs')
('spam', 'eggs')
x = spam
y = eggs
```

Exception Handling

```
try:
   raise Exception('spam', 'eggs')
except Exception, inst:
   print type(inst)     # the exception instance
   print inst.args      # arguments stored in .args
   print inst           # __str__ allows args to printed directly
   x, y = inst          # __getitem__ allows args to be unpacked directly
   print 'x =', x
   print 'y =', y
#output:
<type 'exceptions.Exception'>
('spam', 'eggs')
('spam', 'eggs')
x = spam
y = eggs
```

Generators

```
def reverse(data):
    for index in range(len(data)-1, -1, -1):
        yield data[index]

>>> for char in reverse('golf'):
...     print char
...
f
l
o
g
```

Generators

```
>>> sum(x*y for x,y in zip(xvec, yvec))        # dot product
260

>>> from math import pi, sin
>>> sine_table = dict((x, sin(x*pi/180)) for x in range(0, 91))

>>> unique_words = set(word  for line in page  for word in
line.split())

>>> valedictorian = max((student.gpa, student.name) for
student in graduates)
```

Modules

Every file is a module
filename: test.py

```
import test
test.method(...)        #Like static
x = test.class('test')

from test import *
method(...)
x = class('test')

if __name__ == "__main__":
    pass #When module directly run
else:
    pass #When module is imported
```

Modules

- pyc files are created for imported modules
- pyo files are optimized(!) bytecodes.
- sys.path for PYTHONPATH
- dir(), dir(module), dir(method), ... => dir(object)
- help(object)

Packages

```
sound/                      Top-level package
    __init__.py              Initialize the sound package
    formats/                Subpackage for file format conversions
        __init__.py
        wavread.py
        wavwrite.py
    effects/                Subpackage for sound effects
        __init__.py
        echo.py
        surround.py
    filters/                Subpackage for filters
        __init__.py
        equalizer.py
        vocoder.py
        karaoke.py
```

Packages

```
import sound.effects.echo
sound.effects.echo.echofilter(input, output, delay=0.7, atten=4)

from sound.effects import echo
echo.echofilter(input, output, delay=0.7, atten=4)

from sound.effects.echo import echofilter
echofilter(input, output, delay=0.7, atten=4)

import sound.effects.echo
import sound.effects.surround
from sound.effects import *

from . import echo
from .. import formats
from .. filters import equalizer
```

Standard Library

- Batteries included: A Huge standard library
- 3rd party modules

For more, look for Global Module Index and Python Library Reference on http://docs.python.org

Brief Standard Library Tour...

OS Intervention

- Every command you can use on the shell
- File operations
- User operations
- Subprocess / threading

urllib

```
import urllib
u = urllib.urlopen("http://docs.python.org")
html = u.readlines()
for line in html:
    if '<a href' in line:
        start = line.find('<a href="')
        end = line.find('">')
        print line[start+len('<a href="'):end]
```

tut/tut.html
modindex.html
lib/lib.html
mac/mac.html
ref/ref.html
...

string

```
>>> ",".join(string.letters)
'a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,...,A,B,C,...,Z'
>>> '1,2,3,4,5,6'.split(',')
['1', '2', '3', '4', '5', '6']
>>> '  test  '.strip()
'test'
>>> '  test  '.rstrip()
'  test'
>>> '  test  '.lstrip()
'test  '
>>> string.replace('cemre','e','i')
'cimri'
```

Django

- A high-level Python Web framework that encourages rapid development and clean, pragmatic design.
- DRY Principle (Don't Repeat Yourself)

Django - Models

```
class Reporter(models.Model):
    full_name = models.CharField(max_length=70)

    def __unicode__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateTimeField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = models.ForeignKey(Reporter)

    def __unicode__(self):
        return self.headline
```

Django - Models Usage

```
>>> from mysite.models import Reporter, Article
>>> Reporter.objects.all()
[ ]
>>> r = Reporter(full_name='John Smith')
>>> r.save()
>>> r.id
1
>>> Reporter.objects.all()
[John Smith]
>>> r.full_name
'John Smith'
```

Django - Models Usage

```
>>> Reporter.objects.get(id=1)
John Smith
>>> Reporter.objects.get(full_name__startswith='John')
John Smith
>>> Reporter.objects.get(full_name__contains='mith')
John Smith
>>> r.delete()
```

Django - Urls

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^articles/(\d{4})/$', 'mysite.views.year_archive'),
    (r'^articles/(\d{4})/(\d{2})/$', 'mysite.views.month_archive'),
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'mysite.views.
article_detail'),
)
```

Django - Views

```
def year_archive(request, year):
    a_list = Article.objects.filter(pub_date__year=year)
    return render_to_response('news/year_archive.html',
{'year': year, 'article_list': a_list})
```

Django - Base Template

```
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <img src="sitelogo.gif" alt="Logo" />
    {% block content %}{% endblock %}
</body>
</html>
```

Django - Templates

```
{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
<p>{{ article.headline }}</p>
<p>By {{ article.reporter.full_name }}</p>
<p>Published {{ article.pub_date|date:"F j, Y" }}</p>
{% endfor %}
{% endblock %}
```

Django - Admin Panel

Allows you to achieve Insert/Update/Delete operations for Objects on the database.

Demo

PyQt - Desktop Programming

1. Create a GUI with Qt4-Designer
2. Connect the event (signals) to function slots
3. pyuic4 gui.ui -o gui.py
4. Add these to the end of gui.py:

```
import sys; from gui import *
app = QtGui.QApplication(sys.argv)
window = QtGui.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(window)

window.show()
sys.exit(app.exec_())
```

5. Modify gui.py to use your own signal handler functions

Resources

Python:
http://docs.python.org - Tutorial, Reference, Module Index, etc...
http://www.activestate.com - Interpreter for Windows
http://diveintopython.org - A comprehensive tutorial

Django:
http://www.djangoproject.com
http://www.djangobook.com
http://code.google.com/appengine/docs/

Questions?