Python Introduction

Contact: weber@informatik.uni-hamburg.de
WTM / Informatik Raum F-233
Tel. 42883-2537

Sources: www.csc.villanova.edu/~nlp/python1.ppt
www.cs.umbc.edu/pub/www/courses/graduate/631/Fall2002/Python.ppt
...

Python Features

- interpreted language
- clean syntax, object oriented, powerful extensions
- useful built-in types (lists, dictionaries)

  → for symbolic AI processing
- easy matrix algebra (module `numpy`)

  → for statistical AI processing
- easy to program GUIs
- easy to produce HTML content

Edit and Run a Program

- IDLE integrated development environment (IDE)

  for Linux and commercial operating systems

- `ipython` interactive shell for Linux

  understands python, plus ''magic'' commands:

  `%run file.py` loads and runs a file

  `%hist` prints history of input

- `python file.py` interpretes the file
- `file.py` runs the file, but first make it executable, and

  write inside like: `#!/usr/bin/python`

A Sample of Code ...

```
x = 4 - 3      # comment: integer difference
y = "Hello"

if x == 0 or y == "Hello":
     x = x + 1
     y = y + " World"  # concatenate string

print x
print y
```

Enough to Understand the Code

- assignment uses = and comparison uses ==
- + – * / % compute numbers as expected
- use + for string concatenation
- use % for string formatting
- logical operators are words (and, or, not),
  but not symbols (&&, ||, !)
- first assignment to a variable will create it
- Python assigns the variable types

Basic Data Types

- integers (default for numbers)

  `z = 5 / 2    # answer is 2, integer division`

- floats

  `x = 3.456`

- strings

  Can use `""` or `‘’` to specify. `"abc"` `‘abc’` (same)

  Unmatched quotes can occur within the string. `"matt’s"`

  Use triple double-quotes for multi-line strings or strings which contain both `’` and `"` inside: `"""a‘b"c"""`

Whitespace and Indentation

- use a newline to end a line of code

  (use \ when must go to next line prematurely)

- use consistent indentation to mark blocks of code

Comments

- start comments with # – the rest of line is ignored
- can include a "documentation string" as the first line of any new function or class – the development environment, debugger, and other tools use it: good style to include one

```
def my_function(x, y):
  """This is the docstring. This function
     does blah blah blah."""
# The code would go here ...
```

A Sample of Code ...

```
x = 4 - 3       # comment: integer difference
y = "Hello"

if x == 0 or y == "Hello":
    x = x + 1
    y = y + " World"  # concatenate string

print x
print y
```

Python and Data Types

- Python determines types automatically: "Dynamic Typing"
- But Python is not casual about types, it enforces them thereafter: "Strong Typing"

- E.g., you can't just append an integer to a string.

```
x = "the answer is "      # x is string
y = 23                    # y is integer
print x + y               # Python complains:
TypeError: cannot concatenate 'str' and 'int' objects
```

Naming Rules

- Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.

  `bob   Bob   _bob   _2_bob_   bob_2   BoB`

- There are some reserved words:

  and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while

Multiple Assignment

```
x, y, z = 1, 2, 3
```

```
y
```

```
2
```

map: Apply a Function to an Iterable

```
def heaviside(x):
    if x > 0:
        return 1
    else:
        return 0

a = [-2, -1, 0, 1, 2]
b = map(heaviside, a)
```

```
[0, 0, 0, 1, 1]
```

Many String Operations

- built-in formatting on string data type, e.g.

```
"hello".upper()
 str.upper("hello")        # same

 'HELLO'

"abc;;  456  ".split(";")

 ['abc', '', '  456  ']
```

Printing to Screen

- use % string operator to format output text

```
print "%s xyz %d"  %  ("abc", 34)
```

```
abc xyz 34
```

File Reading and Writing

```
fobj = open("infile.txt", "r")

line = fobj.readline()

rest = fobj.read()

fobj.close()
```

```
fobj = open("outfile.txt", "w")

fobj.write("values: %d, %.6f\n" % (4, 1.234))

fobj.close()
```

Module Import: NumPy

```
import numpy

A = numpy.zeros((2,3))

array([[ 0.,  0.,  0.],

       [ 0.,  0.,  0.]])
```

```
from numpy import random

B = random.uniform(0.0,1.0,(2,3))

array([[ 0.05718388,  0.77822063,  0.85527822],

       [ 0.85458464,  0.98040686,  0.9404561 ]])
```

Assignments are by Reference

- `b = a` does not make a copy; it's the same object

e.g.:

```
a = [1,2,3]
b = a
a.append(7)
print b
```

```
[1,2,3,7]
```

but:

```
a = [1,2,3]
c = a[:]
a.append(7)
print c
```

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

```
import numpy
a = [1,2,3]
d = \
numpy.array(a)
a.append(7)
print d
```

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

NumPy Arrays ≠ Lists

```
A=numpy.array([[1,2],[3,4]])

A[0,1]

  2

A[0,:]

  array([1,2])

A[:,0]

  array([1,3])
```

```
A=[[1,2],[3,4]]

A[0,1]

  TypeError

A[0][1]

  2

A[0][:]

  [1,2]

A[:][0]

  [1,2]     !!
```

Class

```
class stack:
    def __init__(self):
        self.items = []
    def push(self, x):
        self.items.append(x)
    def pop(self):
        x = self.items[-1]
        del self.items[-1]
        return x
    def empty(self):
        return len(self.items) == 0
```

Use like:

```
t = stack()
print t.empty()
 True
t.push("hello")
print t.empty()
 False
t.pop()
print t.empty()
 True
```

Embedding C in Pyhon

```
import scipy.weave
a, b = 1, 2
c = scipy.weave.inline('return_val = a + b;\
printf("a=%d b=%d\\n", a, b);', ['a','b'])
print "c = %d" % c
```

```
a=1 b=2
c = 3
```

Embedding Python in C

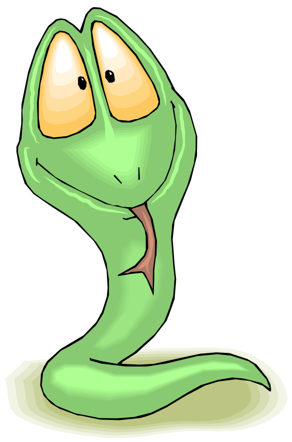
```
 void main () {
   Py_Initialize();
   PyRun_SimpleString("k = \"hello\"");
   PyRun_SimpleString("print k");
   Py_Finalize();
}

gcc embedPythonInC.c -lpython2.6
./a.out
hello
```

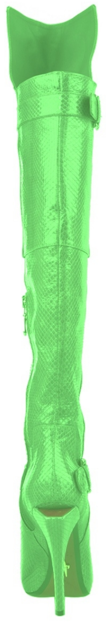
http://www.rexx.com/~dkuhlman/python_book_01.html#extending-and-embedding-python

Links

- http://python.org
- http://www.diveintopython.net
- http://www.rexx.com/~dkuhlman/python_book_01.html
- http://rgruet.free.fr
- http://ipython.org
- http://openbook.galileocomputing.de/python

The End





Python Exercises I/II

- Multiply the two matrices: `C = A B =` $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix}$
- Copy the last two rows of the result `C` into a 2x2 matrix `D`.
- Change a value of `D`. What happens with `C` ?
- What is the difference between: `E=C` and `E=numpy.array(C)` ?
- Which of the following expressions returns the 0th column of `A` ?

  `A[:,0]` or `A[0,:]` or `A[:][0]` or `A[0][:]`
- Append to `A` another column with three times the value -1.

  (to do this, use `numpy.ones` to create a matrix that has a `numpy.shape` of (3,1), and then use `numpy.concatenate`)

Python Exercises II/II

Write a class “stat“. Its constructor creates an empty list.

Its method “sample“ adds its argument (a number) to the list.

Its method “moments“ returns mean and variance of list entries.

Usage of this class is like:

```
s = stat
s.sample(3.0)
mean, sigma = s.moments()
```