Python:
A great programming toolkit

Asokan Pichai
Prabhu Ramachandran

Department of Aerospace Engineering
IIT Bombay

Day 1, Session-1, 10, October 2009

Acknowledgements

This program is conducted by
IIT, Bombay
through CDEEP
as part of the open source initiatives
under the aegis of
National Mission on Education through ICT,
Ministry of HRD.

Outline

About the Workshop

Day 1, Session 1	Sat 09:30–11:00
Day 1, Session 2	Sat 11:15–12:45
Day 1, Session 3	Sat 13:45–15:15
Day 1, Session 4	Sat 15:30–17:00
Day 2, Quiz	Sun 09:00–09:30
Day 2, Session 1	Sun 09:30–11:00
Day 2, Session 2	Sun 11:15–12:45
Day 2, Session 3	Sun 13:45–15:15
Day 2, Session 4	Sun 15:30–17:00

About the Workshop

Intended Audience

- Aimed at Engg., Mathematics and Science teachers.
- Interested students from similar streams.

Goal

Successful participants will be able to use python as their scripting and problem solving language.

Checklist

Live Python

Have you booted using the Live Python DVD?

python

Type python at the command line. Do you see version 2.5 or later?

IPython

Type ipython at the command line. Is it available?

Editor

We recommend scite.

Introduction

- Creator and BDFL: Guido van Rossum
- December 1989
- “Python” as in *Monty Python’s Flying Circus*
- 2.6.x
- PSF license (like BSD: no strings attached)
- Highly cross platform
- Nokia series 60!
- Philosophy: Simple and complete by design

Resources

- Part of many GNU/Linux distributions
- Web: `http://www.python.org`
- Doc: `http://www.python.org/doc`
- Free Tutorials:
  - Official Python tutorial: `http://docs.python.org/tut/tut.html`
  - Byte of Python: `http://www.byteofpython.info/`
  - Dive into Python: `http://diveintopython.org/`

Why Python?

- Readable and easy to use
- High level, interpreted, modular, OO
- Much faster development cycle
- Powerful interactive environment
- Rapid application development
- Rich standard library and modules
- Interfaces well with C++, C and FORTRAN
- More than a math package $\Rightarrow$ some extra work compared to math packages

Use cases

- NASA: Space Shuttle Mission Design
- AstraZeneca: Collaborative Drug Discovery
- ForecastWatch.com: Helps Meteorologists
- Industrial Light & Magic: Runs on Python
- Zope: Commercial grade Toolkit
- Plone: Professional high feature CMS
- RedHat: install scripts, sys-admin tools
- Django: A great web application framework
- Google: A strong python shop

To sum up, python is. . .

- dynamically typed, interpreted $\longrightarrow$ rapid testing/prototyping
- powerful, very high level
- has full introspection
- Did we mention powerful?

But . . .

may be wanting in performance. specialised resources such as SWIG, Cython are available

15 m

Outline

At the prompt, type the following

```
>>> print 'Hello Python'
>>> print 3124 * 126789
>>> 1786 % 12
>>> 3124 * 126789
>>> a = 3124 * 126789
>>> big = 12345678901234567890 ** 3
>>> verybig = big * big * big * big
>>> 12345**6, 12345**67, 12345**678
```

At the prompt, type the following

```
>>> s = 'Hello '
>>> p = 'World'
>>> s + p
>>> s * 12
>>> s * s
>>> s + p * 12, (s + p)* 12
>>> s * 12 + p * 12
>>> 12 * s
```

At the prompt, type the following

```
>>> 17/2
>>> 17/2.0
>>> 17.0/2
>>> 17.0/8.5
>>> int(17/2.0)
>>> float(17/2)
>>> str(17/2.0)
>>> round( 7.5 )
```

Mini exercise

Round a float to the nearest integer, using `int()`?

Midi exercises

- What does this do?
- `round(amount * 10) /10.0`

More exercises?

Round sums

How to round a number to the nearest 5 paise?

Remember $17.23 \longrightarrow 17.25$,

while $17.22 \longrightarrow 17.20$

How to round a number to the nearest 20 paise?

A question of good style

```
amount = 12.68
denom = 0.05
nCoins = round(amount/denom)
rAmount = nCoins * denom
```

Style Rule #1

Naming is 80% of programming

A question of good style

```
amount = 12.68
denom = 0.05
nCoins = round(amount/denom)
rAmount = nCoins * denom
```

Style Rule #1

Naming is 80% of programming

Odds and ends

- Case sensitive
- Dynamically typed ⇒ need not specify a type

```
a = 1
a = 1.1
a = "Now I am a string!"
```

- Comments:

```
a = 1  # In-line comments
# Comment in a line to itself.
a = "# This is not a comment!"
```

30 m

Outline

Basic types

- Numbers: float, int, long, complex
- Strings
- Boolean

Also to be discussed later

tuples, lists, dictionaries, functions, objects. . .

Numbers

```
>>> a = 1 # Int.
>>> l = 1000000L # Long
>>> e = 1.01325e5 # float
>>> f = 3.14159 # float
>>> c = 1+1j # Complex!
>>> print f*c/a
(3.14159+3.14159j)
>>> print c.real, c.imag
1.0 1.0
>>> abs(c)
1.4142135623730951
>>> abs( 8 - 9.5 )
1.5
```

Boolean

```
>>> t = True
>>> f = not t
False
>>> f or t
True
>>> f and t
False
```

Try:

NOT True
not TRUE

Relational and logical operators

```
>>> a, b, c = -1, 0, 1
>>> a == b
False
>>> a <= b
True
>>> a + b != c
True
>>> a < b < c
True
>>> c >= a + b
True
```

Strings

```
s = 'this is a string'
s = 'This one has "quotes" inside!'
s = "I have 'single-quotes' inside!"
l = "A string spanning many lines\
one more line\
yet another"
t = """A triple quoted string does
not need to be escaped at the end and
"can have nested quotes" etc."""
```

More Strings

```
>>> w = "hello"
>>> print w[0] + w[2] + w[-1]
hlo
>>> len(w) # guess what
5
>>> s = u'Unicode strings!'
>>> # Raw strings (note the leading 'r'
... r_s = r'A string $\alpha \nu$'

>>> w[0] = 'H' # Can't do that!
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object does not support item
```

More Strings

```
>>> w = "hello"
>>> print w[0] + w[2] + w[-1]
hlo
>>> len(w) # guess what
5
>>> s = u'Unicode strings!'
>>> # Raw strings (note the leading 'r'
... r_s = r'A string $\alpha \nu$'

>>> w[0] = 'H' # Can't do that!
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object does not support item
```

Let us switch to IPython

Why?

Better help (and a lot more)

Tab completion
?
.?
object.function?

More on strings

```
In [1]: a = 'hello world'
In [2]: a.startswith('hell')
Out[2]: True
In [3]: a.endswith('ld')
Out[3]: True
In [4]: a.upper()
Out[4]: 'HELLO WORLD'
In [5]: a.upper().lower()
Out[5]: 'hello world'
```

Still with strings

```
In [6]: a.split()
Out[6]: ['hello', 'world']
In [7]: ''.join(['a', 'b', 'c'])
Out[7]: 'abc'
In [8] 'd' in ''.join( 'a', 'b', 'c')
Out[8]: False
```

Try:

```
a.split( 'o' )
'x'.join( a.split( 'o' ) )
```

Surprise! strings!!

```
In [11]: x, y = 1, 1.2
In [12]: 'x is %s, y is %s' %(x, y)
Out[12]: 'x is 1, y is 1.234'
```

Try:

```
'x is %d, y is %f' %(x, y)
'x is %3d, y is %4.2f' %(x, y)
```

docs.python.org/lib/typesseq-strings.html

Interlude

A classic problem

How to interchange values of two variables?
Please note that the type of either variable is unknown and it is not necessary that both be of the same type even!

60 m

Outline

Control flow constructs

- `if/elif/else` : branching
- `while` : looping
- `for` : iterating
- `break, continue` : modify loop
- `pass` : syntactic filler

Basic conditional flow

```
In [21]: a = 7
In [22]: b = 8
In [23]: if a > b:
   ....:     print 'Hello'
   ....: else:
   ....:      print 'World'
   ....:
   ....:
World
```

Let us switch to creating a file

Creating python files

- aka scripts
- use your editor
- Note that white space is the way to specify blocks!
- extension `.py`
- run with `python hello.py` at the command line
- in IPython…

If...elif...else example

```
x = int(raw_input("Enter an integer:"))
if x < 0:
     print 'Be positive!'
elif x == 0:
     print 'Zero'
elif x == 1:
     print 'Single'
else:
     print 'More'
```

Simple IO

Console Input

`raw_input()` waits for user input.
Prompt string is optional.
All keystrokes are Strings!
`int()` converts string to int.

Console output

`print` is straight forward. Note the distinction between `print x` and `print x,`

Basic looping

```
# Fibonacci series:
# the sum of two elements
# defines the next
a, b = 0, 1
while b < 10:
    print b,
    a, b = b, a + b
```

```
1 1 2 3 5 8
```

Recall it is easy to write infinite loops with `while`

80 m

So what have we learnt so far?

- The interactive interpreter
- Basic Data Types-Numbers
- `if/elif/else, while`
- Simple IO
- Creating and running a Python script