Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

A not so short introduction to Python

Luca Schenato

Research Institute for Hydrogeological Protection
Italian National Research Council
(CNR-IRPI)

03/10/2011

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

The origin

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Guido Van Rossum, researcher in Amsterdam, was working on an educational language, named ABC, and he came up with this new language: the Python. The language is named after the BBC show “Monty Python's Flying Circus” and has nothing to do with reptiles.

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

Characteristics

Python

L. Schenato

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Python is an object-oriented scripting language. It is as flexible and simple as other scripting languages but it is powerful and rich of functions as standard languages. Python is:

- Free
- Cross-platform
- Fast
- "Garbage-collector" featured
- Easy-to-read & Easy-to-write
- Rich of libraries

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Invoking the Interpreter

Unix

The Python interpreter is usually installed as `/usr/local/bin/python` and can be started by typing the command `python`.

Windows

The Python installation is usually placed in `C:\python27`. To add this directory to your path, type the command `set path=%path%;C:\python27` into the command prompt of a DOS box. The starting command is again `python`.

Invoking the Interpreter

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

The interpreter works like the Unix shell:

- when called with standard input connected to a tty device (dos prompt), it reads and executes commands interactively;
- when called with a file name argument or with a file as standard input, it reads and executes a script from that file. By passing `-i` before the script you enter interactive mode afterwards, otherwise the interpreter exit.

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Invoking the Interpreter

By calling the interpreter it is also possible to execute:

- the statement(s) in command (like shell's -c option) with `python -c command [arg] ...` (it is strongly recommended to quote command with single quotes);
- Python modules as script, as if you had spelled out its full name on the command line, by invoking `python -m module [arg] ...` .

Exiting the interpreter

To exit (with a zero exit status), type an end-of-file character (Ctrl-D on Unix, Ctrl-Z on Windows) at the primary prompt of the interpreter. Alternatively, type `quit()` .

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Argument passing from command line

By executing `import sys` you can access the list of strings `sys.argv` into which the script name and additional arguments are turned: the length of the list is at least one and

- when no script and no arguments are given, `sys.argv[0]` is an empty string;
- when the script name is given as `'-'` (meaning standard input), `sys.argv[0]` is set to `'-'`;
- then `-c` command is used, `sys.argv[0]` is set to `'-c'`;
- when `-m` module is used, `sys.argv[0]` is set to the full name of the located module;
- any other options found after `-c` or `-m` are not consumed by the Python interpreter's option processing but left in `sys.argv`.

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Interactive mode

What does it means

The interpreter is said to be in “interactive mode” when commands
are read from a tty or dos prompt. In this mode, it looks like this:

```
$python
Python 2.5.2 (r252:60911, Jan 24 2010, 14:53:14)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for
     more information.
>>>
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Interactive mode

Multi-line construct requires for continuation lines as in the followinf if statement:

```
>>> python_is_cool = 1
>>> if python_is_cool:
...     print "Man I love Python!"
...
Man I love Python!
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Interactive mode

Multi-line construct requires for continuation lines as in the followinf if statement:

```
>>> python_is_cool = 1
>>> if python_is_cool:
... ␣␣␣␣print "Man I love Python!"
... ↲
Man I love Python!
```

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

Error handling

Errors belongs to two classes:

Handled
: Not real error but "exceptions" handled by an `except` clause in a `try` statement.

Unhandled
: The interpreter prints an error message to the standard error stream and a stack trace. Typing the interrupt character (Control-C or DEL) to the primary prompt cancels the input and returns to the primary prompt (producing a `KeyboardInterrupt` exception).

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Error handling

This is an example

```
>>> python_ic_cool = 1
>>> if python_is_cool
  File "<stdin>", line 1
    if python_is_cool
                    ^
SyntaxError: invalid syntax
>>>
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

How to make a Python script executable

Unix

Putting the line `#! /usr/bin/env python` at the beginning of the script make them executable by the shell. Possibly, make it executable by typing `chmod +x script.py`.

Windows

There is no a corresponding executable mode: any `.py` or `.pyw` file are automatically associated to the python interpreter.

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

How to read the code at the interactive shell

```
>>> # An example
... python_is_cool = 1
>>> if python_is_cool:
...     print "Man I love Python!"
...
Man I love Python!
```

```
>>> # this is the 1st comment
... CNR = 1 # the 2nd comment
>>>           # ... and now a 3rd!
... STRING = "# This is not a comment."
```

How to read the code at the interactive shell

```
>>> # An example A COMMENT
... python_is_cool = 1
>>> if python_is_cool:
...     print "Man I love Python!"
...
Man I love Python!
```

```
>>> # this is the 1st comment
... CNR = 1 # the 2nd comment
>>>         # ... and now a 3rd!
... STRING = "# This is not a comment."
```

How to read the code at the interactive shell

```
>>> # An example A COMMENT
... python_is_cool = 1INPUT LINE
>>> if python_is_cool:INPUT LINE
...     print "Man I love Python!"INPUT LINE
...
Man I love Python!
```

```
>>> # this is the 1st comment
... CNR = 1 # the 2nd comment
>>>         # ... and now a 3rd!
... STRING = "# This is not a comment."
```

Outline
The origin
Python's characteristics
Python Interpreter
Python Environment
A very basic use of Python
Credit

How to read the code at the interactive shell

```
>>> # An example A COMMENT
... python_is_cool = 1INPUT LINE
>>> if python_is_cool:INPUT LINE
...     print "Man I love Python!"INPUT LINE
... SECONDARY PROMPT OF INPUT LINE
Man I love Python!
```

```
>>> # this is the 1st comment
... CNR = 1 # the 2nd comment
>>>         # ... and now a 3rd!
... STRING = "# This is not a comment."
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

How to read the code at the interactive shell

```
>>> # An example A COMMENT
... python_is_cool = 1INPUT LINE
>>> if python_is_cool:INPUT LINE
...     print "Man I love Python!"INPUT LINE
... SECONDARY PROMPT OF INPUT LINE
Man I love Python!OUTPUT LINE
```

```
>>> # this is the 1st comment
... CNR = 1 # the 2nd comment
>>>         # ... and now a 3rd!
... STRING = "# This is not a comment."
```

Python as a calculator

The interpreter is also a simple calculator: you can type an expression and it will compute the value. Expression syntax is straightforward with the operators +, -, * and / that works just like in most other languages; parentheses ( ) can be used for grouping. Let's try:

```
>>> 1+32
33
>>> # This is a comment
... 1+32
33
>>> 1+32   # and a comment on the same line as code
4
>>> (20-3*4)/6
2
```

Python as a calculator: implicit casting

Please note the following example:

```
>>> # Division of floating points give a double
... 10.0/3.0
3.3333333333333335
>>> # Division of integers gives an integer (the floor):
... 10/3
3
>>> # Division of integer and floating point give a
      floating point
... 10.0/3
3.3333333333333335
>>> 10/3.0
3.3333333333333335
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Python as a calculator: variable assignment

To assign a value to a variable use the equal sign ('='). No result is displayed before the next interactive prompt:

```
>>> a = 10
>>> b = 3*4
>>> a * b
120
```

Multiple assignment is allowed (like in C):

```
>>> a = b = c = 0   # Zero a, b and c
>>> a
0
>>> b
0
>>> c
0
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Python as a calculator: variable assignment

Before using a variable, it
has to be defined (i.e. assigned to a value) or an error will occur:

```
>>> # trying to access an undefined variable
... x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
```

Python as a calculator: numerical types

Floating points are fully supported and operators with mixed type operands convert the integer operand to floating point:

```
>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2
3.5
>>> 2.5 ** 2
6.25
>> _
6.25
>> _ * 2
12.50
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Python as a calculator: numerical types

Floating points are fully supported and operators with mixed type operands convert the integer operand to floating point:

```
>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2
3.5
>>> 2.5 ** 2 Power operator
6.25
>> _
6.25
>> _ * 2
12.50
```

Python as a calculator: numerical types

Python

L. Schenato

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Floating points are fully supported and operators with mixed type operands convert the integer operand to floating point:

```
>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2
3.5
>>> 2.5 ** 2 Power operator
6.25
>> _ last printed expression
6.25
>> _ * 2
12.50
```

Python as a calculator: numerical types

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Complex numbers are supported in Python; imaginary numbers are written with a suffix of `j` or `J`. Complex numbers with a nonzero real component are written as `(real+imagj)`, or can be created with the `complex(real, imag)` function (like in Matlab©).

```
>>> 1j * 1J
(-1+0j)
>>> 1j * complex(0,1)
(-1+0j)
>>> (3+1j)*3
(9+3j)
>>> (1+2j)/(1+1j)
(1.5+0.5j)
```

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Python as a calculator: numerical types

Complex numbers are always represented as two floating point numbers, the real and imaginary part. To extract these parts from a complex number z, use `z.real` and `z.imag`.

```
>>> a=0.5+2.5j
>>> a.real
0.5
>>> a.imag
2.5
```

Use `abs(z)` to get its magnitude (as a float) or `z.real` to get its real part (conversion functions to floating point and integer, i.e. `float()`, `int()` and `long()`, don't work for complex numbers.

Python as a calculator: numerical types

An example:

```
>>> a=3.0+4.0j
>>> float(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: can't convert complex to float; use abs(z)
>>> a.real
3.0
>>> a.imag
4.0
>>> abs(a)   # sqrt(a.real**2 + a.imag**2)
5.0
```

Strings

Python strings can be expressed in several ways. They can be enclosed in single quotes or double quotes:

```
>>> 'spam eggs'
'spam eggs'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

Strings

String literals can span multiple lines in several ways. Continuation lines can be used, with a backslash as the last character on the line indicating that the next line is a logical continuation of the line:

```
>>> hello = "This is a rather long string containing\n\
... several lines of text just as you would do in C.\n\
...    Whitespace is\
... significant."
>>>
>>> print hello
This is a rather long string containing
several lines of text just as you would do in C.
   Whitespace is significant.
```

Eventually, strings can be surrounded in a pair of matching triple-quotes: `"""` or `’’’`: end of lines can be omitted in this case.

Strings

A string literal can be made a "raw" string: in this case, `\n` sequences are not converted to newlines, but the backslash at the end of the line, and the newline character in the source, are both included in the string as data.

```
>>> hello = r"This is a rather long string containing\n\
... several lines of text much as you would do in C."
>>> print hello
This is a rather long string containing\n\
several lines of text much as you would do in C.
```

Strings

Strings
can be concatenated with the `+` operator, and repeated with `*`:

```
>>> word = 'CNR' + '-' + 'IRPI'
>>> word
'CNR-IRPI'
>>> '<' + word*5 + '>'
'<CNR-IRPICNR-IRPICNR-IRPICNR-IRPICNR-IRPI>'
```

Two string literals next to each other are automatically concatenated:

```
>>> word = 'CNR' '-' 'IRPI'
>>> word
'CNR-IRPI'
```

Length of a string can be obtained by the built-in function `len()`.

Python

L. Schenato

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Strings

Indexing of strings are allowed and like in C, the first character of a string has index 0. In python there is no character type, but a character is simply a one-sized string. Substrings can be specified with the slice notation: two indices separated by a colon (like in Matlab©).

```
>>> word[4]
'I'
>>> word[0:2]
'CN'
>>> word[2:4]
'R-'
```

An omitted first index defaults to zero, an omitted second index defaults to the size of the string being sliced.

```
>>> word[:2]     # The first two characters
'CN'
>>> word[2:]     # Everything except the first two
    characters
'R-IRPI'
```

Outline
The origin
Python's characteristics
Python Interpreter
Python Environment
A very basic use of Python
Credit

Strings

Python strings are not like C string: they cannot be changed, as is the following example:

```
>>> word[0] = 'c'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> word[:1] = 'cnr'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Please, try the following code: `word[1:100]`, `word[10:]`, `word[2:1]`, `word[-1]`, `word[-2]`, `word[-2:]`, `word[:-2]`, `word[-0]`, `word[-10]`,...

Data Types

Type	Internal representation	Example
Integer	32 bit (“C” long int)	1220, 0, -15
Long integer	more than 32 bit	10000000L, -1243574L
Float	32 bit (“C” double)	1.23 2.32e-9, 5.0E201
Boolean	integer	0, 1
Complex	couple of float	1+2j, 4.0+1.1j, 2j
String	array of characters	'CNR', "l'acqua"

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Standard numeric operators

Operator	Description	Example
+,-	Sum, Difference	1+2=3, 3-4=-1
*,/	Multiplication, Division	4*3=12, 10/5=2
**	Power	2**3=8,3**0.5=1.73
%	Modulo operation	10%3=1, 5.3%2.5=0.3
«,»	bitwise left-,right-shift	15«1=30, 18»1=9

Standard boolean operators

Operator	Description	Example
or, and	logic or, and	x or y, z and k
not	logic not	(not 0)=1
<, <=, >, >= ==, <>, !=	comparison	(10==10)=1, ('a'!='a')=0
|	bit-or	x | y
&	bit-and	x& y
^	bit-xor	x^y

Standard string operators

Operator	Description	Example
+	cat	('a'+'b')='ab'
*	repetition	('a'*3)='aaa'
s[i]	indexing	s='abc',s[0]='a'
s[i:j]	slicing	s='abc',s[1:2]='b'
len(s)	length	s='abc',len(s)=3
%	formatting	'hi %s' % 'luca'='hi luca'

Parameter	Description
%s	stringa
%c	single char string
%d	number
%u	unsigned integer
%o	octal number
%x	hex number
%g	float
%e	float, scientific notation

1 The origin

2 Python's characteristics

3 Using the Python Interpreter

4 The Environment of the Python Interpreter

5 A very basic use of Python

6 Credit

Outline

The origin

Python's characteristics

Python Interpreter

Python Environment

A very basic use of Python

Credit

Credit

Credit goes to www.python.org and herein contents.