

Curso de Python

Introducción al lenguaje

Carlos Hernando

`chernando@acm.org`

ACM Facultad de Informática
Universidad Politécnica de Madrid

13 de noviembre de 2007
Curso Python 2007



Contenido

- 1 Introducción
- 2 Primeros pasos
 - El intérprete
 - Tipos básicos
 - Listas
 - Control de flujo
- 3 Subiendo a cosas más serias
 - Ficheros .py y módulos
 - Entrada y salida
- 4 Estructuras de datos
 - Listas, de nuevo
 - Tuplas
 - Diccionarios
 - Conjuntos

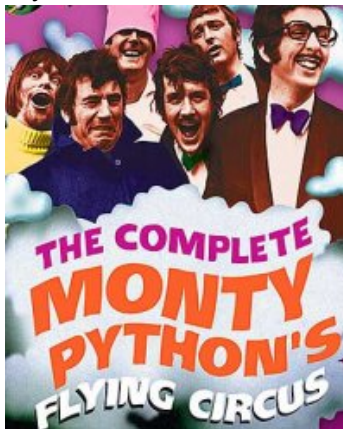
El lenguaje

Principales características del lenguaje:

- Es un lenguaje desarrollo **rápido**.
- Especial para perezosos.
- Es un lenguaje serio.
- Orientado a objetos.
- Multiplataforma.
- Dispone de una API muy extensa.
- ...

Cosas a recordar

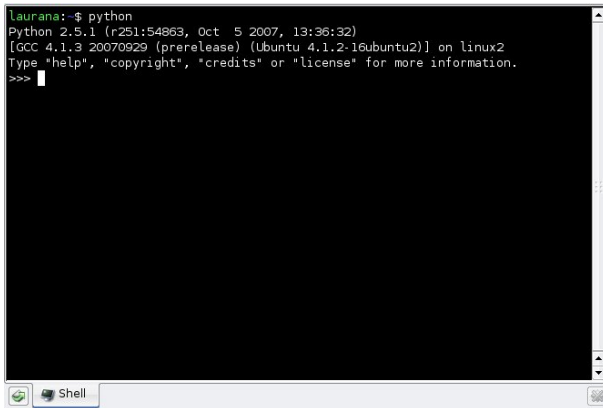
- Existen varias versiones, aquí hablaremos de la 2.5.
- Python no es de pitón.
- Es de los Monty Python.



Python es interpretado

Lanzamiento

python



```
Laurana:~$ python
Python 2.5.1 (r251:54863, Oct 5 2007, 13:36:32)
[GCC 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

The image shows a terminal window titled "Shell" with a taskbar icon. The terminal output displays the Python version (2.5.1), build information (r251:54863, Oct 5 2007, 13:36:32), and the compiler (GCC 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)) on a Linux system. It prompts the user to type "help", "copyright", "credits", or "license" for more information. The prompt ">>>" is followed by a cursor and a vertical bar.

Una calculadora normal y corriente

Operaciones básicas

```
>>> 1 + 1
2
>>> 2 * 2
4
>>> 6 / 3
2
>>> 2**10
1024
```

Funciones matemáticas

```
>>> import math
>>> math.sqrt(4)
2.0
>>> math.log10(100)
2.0
>>> math.pi
3.1415926535897931
```

Variables

- Se declaran al vuelo.
- Evidentemente es de tipado débil.
- Cuidado con los errores tipográficos.

Example

```
>>> a = b = c = 1
>>> a = 'ha!'
>>> a, b, c
('ha!', 1, 1)
```

Example

```
>>> hola = 1
>>> ohla = "hey"
>>> hola
1
```

Números

Funciones de números

```
>>> float(1)
1.0
>>> int(1.0)
1
>>> abs(-10)
10
```

Números complejos

```
>>> (1 + 1j) + 2j
(1+3j)
>>> a = (1 + 2j)
>>> a.imag, a.real
(2.0, 1.0)
```

Conversiones de tipo

$3 / 2 = 1$

$3.0 / 2 = 1.5$

Cadenas

Declaración

```
>>> c1 = 'cadena1'  
>>> c2 = "cadena2"  
>>> c3 = """  
...     cadena  3  
...     """  
>>> c3  
'\n\tcadena\t3\n'
```

Ejemplos de uso

```
>>> a = 'foo'  
>>> a = a + ' bar'  
>>> a  
'foo bar'  
>>> a[0]  
'f'  
>>> b = 'b' + a[1] * 4
```

Comienza la fiesta

Python se define por sus listas

Example

```
>>> lista = []
>>> lista
[]
>>> lista.append(1)
>>> lista.append(2)
>>> lista
[1, 2]
>>> len(lista)
2
```

Example

```
>>> lista.reverse()
>>> lista
[2, 1]
>>> lista.remove(1)
>>> lista
[2]
>>> lista.count(2)
1
```

Preguntas

Example

```
lista = [1, 2.0, "hey!", []]  
lista[1]  
lista[0] += 2  
lista.index(2.0)  
lista.insert(3,2)  
lista.count(2)  
lista.index(2.0)  
lista.index([]).append([4, 5, 6])  
lista[4].append(lista)  
lista[-2]  
lista[:2]  
lista[2:]
```

if

Prototipo

```
if condicion:
    verdad_1
elif condicion2:
    verdad_2
else:
    falso
```

Example

```
>>> operador = '+'
>>> if operador == '-':
...     print "Resta"
... elif operador == '+':
...     print "Suma"
... else:
...     print "eing?"
...
Suma
```

switch

No hay.

for

La idea es siempre iterar, como un foreach.

Protipo

```
for elemento in secuencia:  
    cuerpo
```

Example

```
for e in lista:  
    print e
```

Ampliando for

Oye, yo quiero hacer un for de toda la vida...

Una secuencia numérica:

Example

```
for i in range(0,10):  
    print i
```

Ahora sobre una lista:

Example

```
for n, e in enumerate(lista):  
    print n, e
```

while

Protipo

```
while condicion:  
    cuerpo
```

Example

```
>>> a = 0  
>>> while a < 10:  
...     print a  
...     a += 1
```

Funciones

Prototipo

```
def funcion(args):  
    cuerpo  
    return valor
```

Example

```
>>> def funcion(a, b):  
...     resultado = 0  
...     resultado += a  
...     resultado += b  
...     return resultado  
...  
>>> funcion(2, 2)
```

Final de la primera parte

Dudas, cuestiones, comentarios... pues a descansar.



No todo es el intérprete

- Los ficheros para python terminan **.py**
- Para ejecutarlos en GNU/Linux:

```
#!/usr/bin/env python
```


y fijar los permisos adecuados.
- Para ejecutarlo en Windows “sin” intérprete tenemos:
<http://www.py2exe.org/>

Organizando nuestros programas

HEY! Aquí te falta contenido!



Por pantalla

- Para mostrar datos: print

Example

```
print "Hola %s" % (nombre, )  
print "%(id)s %(usuario)s", % dict
```

- Para pedir datos: raw_input

Example

```
numero = raw_input('Dame un numero ')
```

Ficheros

- Abrir un fichero: `open(ruta, modo)`
- Para leer tenemos varias opciones:

Example

```
t = f.read()
l = f.readline()
ls = f.readlines()
```

- Para escribir: `write(objeto)`
- Para cerrar: `close()`
- Para moverse: `seek(posición)`

Serializar

- Necesitamos a pickle
- Para guardarlo: `pickle.dump(objeto, fichero)`
- Para recuperarlo: `objeto = pickle.load()`

En realidad no estoy hablando de ficheros.

Pilas

Las listas pueden comportarse como pilas, simplemente utilizamos el método `pop()`.

Example

```
>>> pila.append(1)
>>> pila.append(2)
>>> pila.pop()
2
>>> pila.append(3)
>>> pila.pop()
3
>>> pila.pop()
1
>>> pila
[]
```

Colas

Al igual que el caso de las pilas podemos hacer uso de las listas como colas con `pop(0)`

Example

```
>>> cola = []
>>> cola.append(1)
>>> cola.append(2)
>>> cola.pop(0)
1
>>> cola.append(3)
>>> cola.pop(0)
2
>>> cola[]
[3]
```

No me toques la lista

Las tuplas son listas **inmutables**

Example

```
>>> tupla = ("Curso Python", 23, "Carlos Hernando")
>>> tupla
('Curso Python', 23, 'Carlos Hernando')
>>> tupla[1]
23
>>> tupla.append('chernando@acm.org')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no...
>>> tupla[1] += 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not...
```

Buscando las cosas por su nombre

Es como una lista, pero con identificador asociados:

Example

```
>>> diccionario = {}
>>> diccionario['Curso'] = 'Curso Python'
>>> diccionario['Charlatan'] = 'Carlos Hernando'
>>> diccionario
{'Curso': 'Curso Python',
 'Charlatan': 'Carlos Hernando'}
>>> diccionario['Asistentes'] = 23
>>> diccionario['Asistentes'] += 2
>>> diccionario
{'Curso': 'Curso Python',
 'Charlatan': 'Carlos Hernando',
 'Asistentes': 25}
```

Conjuntos

Example

```
>>> a = set([ 2, 3, 2 ])
>>> b = set([ 1, 5, 3 ])
>>> a & b
set([3])
>>> a - b
set([2])
>>> b - a
set([1, 5])
>>> a ^ b
set([1, 2, 5])
```

Resumen

- Conocer el intérprete.
- Defenderse con la sintaxis.
- Manejar los tipos básicos y no tan básicos.
- Utilizar algunos módulos.
- Descubrir en Python un gran lenguaje ;-)

Gracias a todos

