

Python: Primeros Pasos

Renato Covarrubias Romero
`rcovarru@inf.utfsm.cl`

Duoc Linux Day
Comunidad Linux Duoc



viernes, 14 de agosto de 2009

Contenido

- 1 **Introducción**
- 2 **Tipos de Datos**
- 3 **Funciones Base**
- 4 **Clases**

Contenido

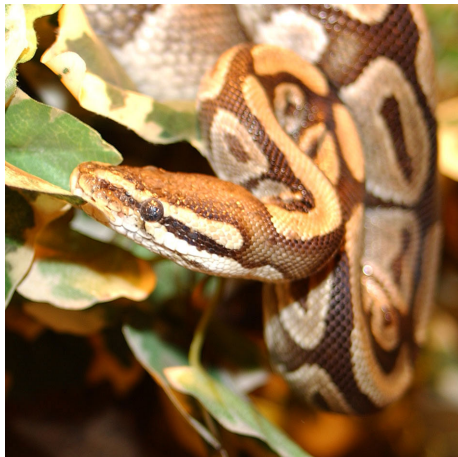
1 Introducción

¿Qué es python?

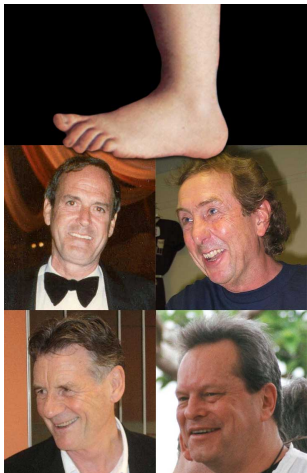
- ▶ Es un proyecto de Código Abierto
- ▶ Es un lenguaje de programación interpretado multiparadigma:
 - ▶ Es imperativo (como C o Basic)
 - ▶ Es orientado a objetos (como C++ o Java)
 - ▶ Es funcional (como Lisp, Haskell o Scheme)
 - ▶ etc...
- ▶ Tipificado débil.
- ▶ Es el “*antónimo*” a Perl, python es más limpio y elegante para programar.

¿De donde viene el nombre?

Lamentablemente no de la familia de serpientes constrictoras.



¿De donde viene el nombre?



Guido van Rossum (BDFL, Benevolent Dictator for Life), llamó al lenguaje en honor al grupo británico de humoristas los Monty Python (a veces conocidos como los Pythons).

Filosofía

- 1 Bello es mejor que feo.
- 2 Explícito es mejor que implícito.
- 3 Simple es mejor que complejo.
- 4 Complejo es mejor que complicado.
- 5 Plano es mejor que anidado.

Filosofía

- 6 Ralo es mejor que denso.
- 7 La legibilidad cuenta.
- 8 Los casos especiales no son tan especiales como para quebrantar las reglas.
 - 1 Aunque lo práctico gana a la pureza.
- 9 Los errores nunca deberían dejarse pasar silenciosamente.
 - 1 A menos que hayan sido silenciados explícitamente.
- 10 Frente a la ambigüedad, rechaza la tentación de adivinar.

Filosofía

- 11 Debería haber una (y preferiblemente sólo una) manera obvia de hacerlo.
 - 1 Aunque esa manera puede no ser obvia al principio a menos que usted sea Holandés.
- 12 Ahora es mejor que nunca.
 - 1 Aunque nunca es a menudo mejor que ahora ya.
- 13 Si la implementación es difícil de explicar, es una mala idea.
- 14 Si la implementación es fácil de explicar, puede que sea una buena idea.
- 15 Los espacios de nombres (namespaces) son una gran idea ¡Hagamos más de esas cosas!

Filosofía

Se puede encontrar un “huevo de pascua” con la filosofía.

```
1 import this
```

Contenido

2 Tipos de Datos

Número Entero

int

Precisión fija

```
1 a = 42  
2 print a
```

Número Entero

long

Precisión arbitraria

```
1 # 261 - 1  
2 a = 2305843009213693951L  
3 print a
```

Número

float

Punto/Coma Flotante

```
1 # pi
2 a = 3.1415926535897931
3 print a
```

Booleano

bool

Verdadero o Falso

```
1 # Morphy  
2 a = False  
3 print a
```

String

str

Inmutable

```
1 a = "Hola_Mundo"  
2 print a  
3 print a+1  
4 print a*2
```

String

unicode

Versión unicode de str

```
1 a = u'\xd1and\xfa'  
2 print a
```

Secuencia

list

Mutable, puede contener diversos tipos

```
1 lista = ['aaa', 1, 90]
2 print lista[-1]
3 lista[0] = 'bbb'
4 print lista[0:2]
```

Secuencia

tuple

Inmutable

```
1 tupla = (1, 2, 3)
2 tupla[0] = 2
3 tupla[0]
4 otratupla = (tupla, ('a', 'b'))
```

Mapping

dict

Grupo de pares claves, valor

```
1 dict = {"dia": 24, "mes": "agosto"}  
2 for k in dict:  
3     print "%s=%s" %(k, dict[k])
```

Conjunto

set y frozenset

sin orden, no contiene duplicados.

```
1 conj = set(['a', 'b', 'a'])  
2 print conj
```

Falso?

¿Con tanto tipo de dato, que valores se consideran verdades y cuales falsos?

- ▶ None
- ▶ False
- ▶ 0, OL, 0.0, 0j
- ▶ '', (), [], {}
- ▶ Para las clases, cuando `__nonzero__()` o `__len__()` retornan 0 o False.

Contenido

3 Funciones Base

(Algunas) Funciones Base

- ▶ `abs(x)`
- ▶ `all(i)`, `any(i)`
- ▶ `bool([x])`
- ▶ `chr(i)`, `ord(c)`, `unichr(i)`
- ▶ `complex([real[, imag]])`
- ▶ `dict([arg])`
- ▶ `divmod(a,b)`
- ▶ `filter(funcion, iterable)`
- ▶ `map(funcion, iterable [, ...])`
- ▶ `reduce(funcion, iterable [, iniciador])`
- ▶ `sum(iterable [, inicio])`
- ▶ `float([x])`
- ▶ `help([objeto])`
- ▶ `hex(x)`

(Algunas) Funciones Base

- ▶ `input ([prompt]), raw_input ([prompt])`
- ▶ `int ([x[, radix]])`
- ▶ `len(s)`
- ▶ `list ([iterable])`
- ▶ `long ([x[, radix]])`
- ▶ `min(iterable [...][key])`
`max(iterable [...][key])`
- ▶ `oct(x)`
- ▶ `open(nombre[,modo[,buf]])`
- ▶ `pow(x, y[,z])`
- ▶ `range([inicio,] fin [, salto])`
`xrange([inicio,] fin [, salto])`
- ▶ `round(x[,n])`
- ▶ `tuple ([iterable])`

Funciones anónimas

Heredado de los lenguajes funcionales, existe `lambda` para definir funciones anónimas.

```
1 f = lambda x: x+1
2 print f(4)
```

Operadores Lógicos

- ▶ `x or y`
- ▶ `x and y`
- ▶ `not x`

`not` tiene la menor prioridad en operadores booleanos. `not a==b` es interpretado como `not (a==b)`.
`a == not b` causará un error de sintaxis.

Comparaciones

- ▶ <
- ▶ <=
- ▶ >
- ▶ >=
- ▶ ==
- ▶ != 0 <>
- ▶ *is*
- ▶ *is not*

Operaciones Numéricas

int, float, long, complex

- ▶ $x + y$
- ▶ $x - y$
- ▶ $x * y$
- ▶ x / y
- ▶ $x \% y$
- ▶ $-x$
- ▶ $+x$
- ▶ $x ** y$

Operaciones al bit

- ▶ $x|y$
- ▶ $x \wedge y$
- ▶ $x \& y$
- ▶ $x \ll n$
- ▶ $x \gg n$
- ▶ $\sim x$

Tipos de Secuencia

str, unicode, list, tuple, ...

Los string puede ser con 'comilla simple' o "comilla doble".

- ▶ `x in s`
- ▶ `x not in s`
- ▶ `s + t`
- ▶ `s*n, n*s`
- ▶ `s[i]`
- ▶ `s[i:j]`
- ▶ `s[i:j:k]`
- ▶ `len(s), min(s), max(s)`

(Algunos) Métodos de Strings

- ▶ `capitalize()`, `lower()`, `upper()`, `swapcase()`
- ▶ `center(ancho[,caracter])`
- ▶ `count(sub[,inicio[,fin]])`
- ▶ `find(sub[,inicio[,fin]])`
- ▶ `isalpha()`, `isdigit()`, `islower()`,...
- ▶ `join(seq)`, `split([sep[,max]])`
- ▶ `replace(old, new[,count])`
- ▶ `zfill(ancho)`

Métodos Secuencias Mutables

- ▶ `s[i] = x`
- ▶ `s[i:j] = t`
`del s[i:j]`
- ▶ `s[i:j:k] = t`
`del s[i:j:k]`
- ▶ `s.append(x)`
`s.insert(i,x)`
`s.pop([i])`
- ▶ `s.count(x)`
- ▶ `s.index(x[, i [, j]])`
- ▶ `s.insert(i,x)`
`s.remove(x)`
- ▶ `s.reverse()`
- ▶ `s.sort([cmp[,key[,rev]])`

(Algunos) Métodos Diccionarios

- ▶ `a[k]`
- ▶ `a[k] = v`
`del a[k]`
- ▶ `k in a`
`a.has_key(k)`
- ▶ `a.items()`
`a.keys()`
`a.values()`
- ▶ `a.clear()`

Contenido

4 Clases

¿Cómo se define una clase?

```
1 class Charla:
2     a = 'Python'
3     def __init__(self):
4         self.b = "Primeros Pasos"
5     def nombre(self):
6         return self.a+": "+self.b
7
8 a = Charla()
9 print a.nombre()
```

¿Existen clases definidas?

!Por supuesto! :D y están agrupadas en módulos.

```
import modulo
```

```
1 import datetime  
2 print(datetime.datetime.now())
```

¿Existen clases definidas?

Son demasiadas, y hay para muchas cosas. La documentación la pueden encontrar en:

<http://docs.python.org/library/>

10^{mo} Encuentro Linux

2009.encuentrolinux.cl

22, 23 y 24 de octubre.

¿Consultas?