Python para RPi

Parte II: Introducción a Python

Manel Velasco,[1] PhD and Alexandre Perera,[1,2] PhD

[1]Departament d'Enginyeria de Sistemes, Automatica i Informatica Industrial (ESAII)
Universitat Politecnica de Catalunya

[2]Centro de Investigacion Biomedica en Red en Bioingenieria, Biomateriales y Nanomedicina (CIBER-BBN)
Alexandre.Perera@upc.edu Manel.Velasco@upc.edu

Introduction to Python and Raspberry Pi
July, 2013

Contents I

¿Que es Python?

Hola

Ejemplo

este es un ejemplo

Why not?

- ¿Python?¿Python?¿Python?¿Python?
  - Ventajas:
    - Muchas librerias científicas en el mercado
    - Lenguaje bien pensado desde un principio, muy facil de leer y bien estructurado, "Codificamos lo que pensamos".
    - Muchas librerias para aspectos no relacionados con la ciencia (servidores web, captura de imágenes...)
    - Libre y abierto, muy extendido y con una comunidad muy activa.
  - Desventajas:
    - Los entornos de desarrollo son menos placenteros que en otros lenguajes.
    - Desde un punto de vista científico no es posible encontrar todos los algoritmos implemenetados.

No es obligatorio

No hace falta usar Python... pero...,
No pasa nada por intentarlo

Historia

Historia

- Python 1.0 - Enero 1994
  - Python 1.5 - Diciembre 31, 1997
  - Python 1.6 - Septiembre 5, 2000
- Python 2.0 - Octubre 16, 2000
  - Python 2.1 - Abril 17, 2001
  - Python 2.2 - Diciembre 21, 2001
  - Python 2.3 - Julio 29, 2003
  - Python 2.4 - Noviembre 30, 2004
  - Python 2.5 - Septiembre 19, 2006
  - Python 2.6 - Octubre 1, 2008
  - **Python 2.7 - Julio 3, 2010**
- Python 3.0 - Diciembre 3, 2008
  - Python 3.1 - Junio 27, 2009
  - Python 3.2 - Febrero 20, 2011
  - Python 3.3 - Septiembre 29, 2012

Guido van Rossum

Pensado al final de los 80s por

Installation

Linux

apt-get install python

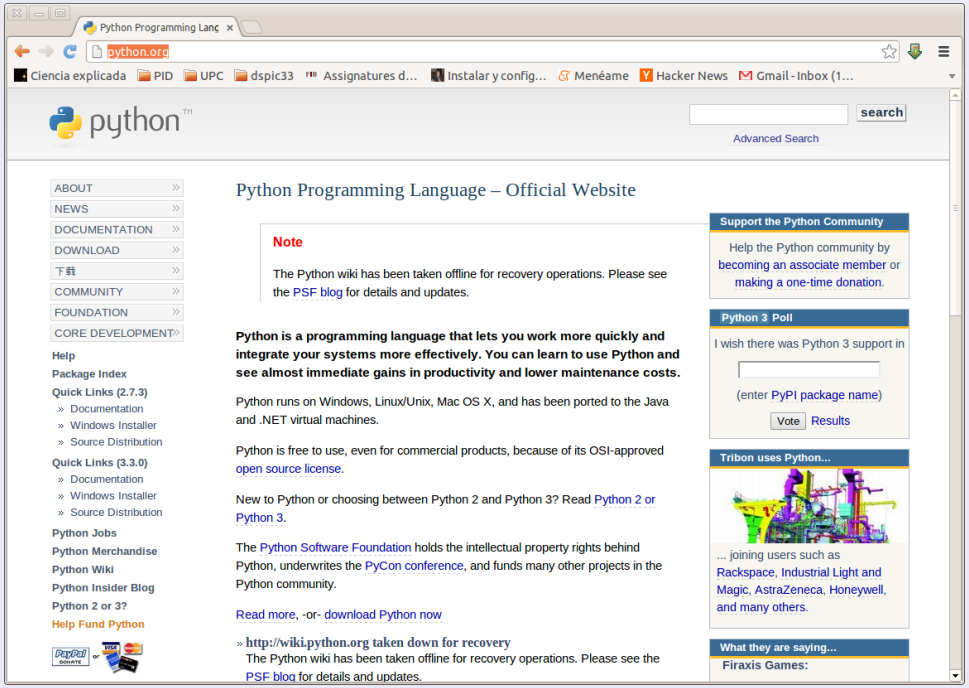
Windows

Ir a **http://www.python.org/getit/** y descargar **Python 2.7.3 Windows Installer**

Resources

AYUUUUUUUUDDDAAAAAAAA!!!

http://python.org

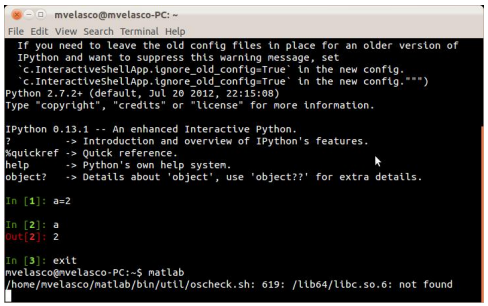


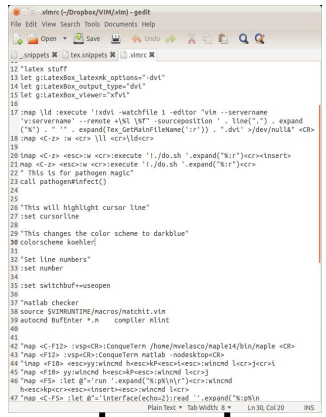
Modos de trabajar

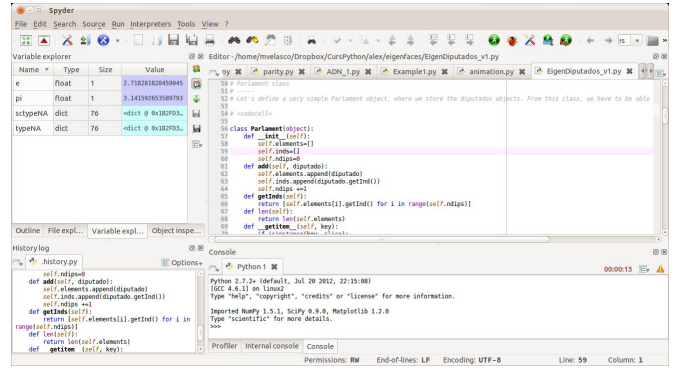
IDE

Script

Python Shell







Python core

El intérprete Python

El intérprete Python

Python es abierto, es una especificación, por lo tanto hay muchas implementaciones de Python:

CPython	Implementación or defecto (C, C++)
CLPython	Implementación de Python en Lisp
Jython	Implementación de Python en Java
PyPy	La implementación de Python en Python
IronPython	Implementación en C#

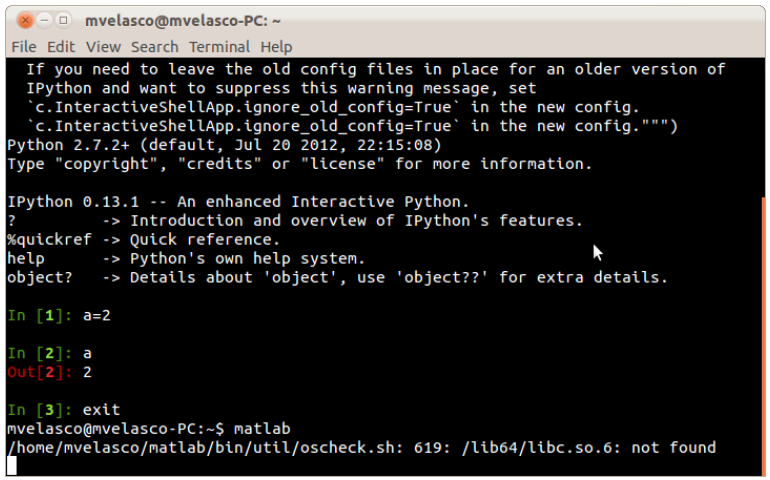
Intérprete Python

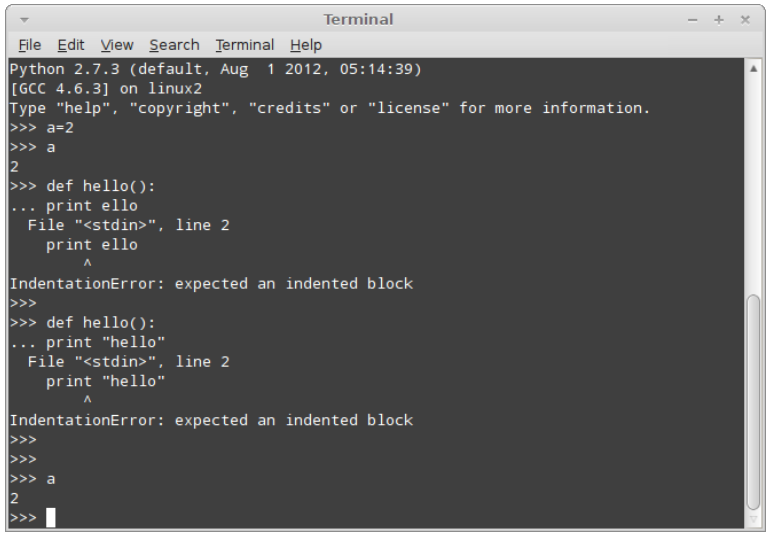
Python Shell

Python Shell

Hay muchas maneras d trabajar directamente con Python, las más destacadas son:

- **CLIPython** La manaera más sencilla.
- **IPython** shell de python mejorada (muy mejorada)





Editores de texto

Editores de texto

Cualquier editor de textos es adecuado para escribir programas en Python, pero recomendamos que tenga algunas características:

- Substitución de tabuladores por espacios.
- Inserción automática de pedazos de código repetitivos
- Autocompletado

En el universo Linux, tanto Vim como Emacs tienen estas características.

IDEs

Los IDEs más adecuados

Spyder Entorno parecido a Matlab, orientado a entornos científicos.

Eclipse-PyDEV IDE orientado a grandes proyectos

DEMO

Primer paso

Paso 1

Abra un intérprete y escriba ...

```
>>> print "Hello, world"
Hello, world
```

Bienvenido a Python,
acaba de ejecutar su primera instrucción en python, felicidades!

Segundo paso

Paso 2

Para emepezar a sentirnos cómodos en python inroduzca la siguiente ristra de instrucciones en el intérprete

```
>>> a = 3
>>> b = 2*a
>>> type(b)
<type 'int'>
>>> print b
6
>>> a*b
18
>>> b = 'hello'
>>> type(b)
<type 'str'>
>>> b + b
'hellohello'
>>> 2*b
'hellohello'
```

Second step

STEP 2

Para emepezar a sentirnos cómodos en python inroduzca la siguiente ristra de instrucciones en el intérprete

```
>>> a = 3
>>> b = 2*a
>>> type(b)
<type 'int'>
>>> print b
6
>>> a*b
18
>>> b = 'hello'
>>> type(b)
<type 'str'>
>>> b + b
'hellohello'
>>> 2*b
'hellohello'
```

Observe que...

- No declaramos variables (hurra!!!!!)
- El tipo de la variable puede cambiar cuando queramos (hurra!!!, hurra!!!)
- Hay una manera de saber cual es el tipo de una variable en un momento dado.

Types

Integer

```
>>> 1+1
2
>>> a=4
```

Float

```
>>> c=2.1
>>> 3.5/c
1.6666666666666665
```

Boolean

```
>>> 3 > 4
False
>>> test = (3 > 4)
>>> test
False
>>> type(test)
<type 'bool'>
```

Complex

```
>>> a=1.5+0.5j
>>> a.real
1.5
>>> a.imag
0.5
>>> import cmath
>>> cmath.phase(a)
0.3217505543966422
```

Calculadora Básica

Un shell de Python es como una calculadora de bolsillo con las operaciones básicas +, -, *, /, % (modulo) implementadas de forma nativa:

```
>>> 7 * 3.
21.0
>>> 2**10
1024
>>> 8 % 3
2
```

ATENCIÓN!

División entera

```
>>> 3/2
1
```

Utilice floats

```
>>> 3 / 2.
1.5
>>> a = 3
>>> b = 2
>>> a / b
1
>>> a / float(b)
1.5
```

EJERCICIOS BÁSICOS

Escriba scripts que resuelvan los siguientes problemas

- ¿Cuál es la diferencia entre usar "+" y "", en un comando de impresión? Pruébelo!
- Escriba un programa que pregunte al usuario dos nombres, guarde estos nombres en variables y escriba un saludo a ambos a la vez

Listas

Python proporciona un conjunto de tipos básicos denominados "contenedores" que son muy importantes dentro de la estructura propia del lenguaje

Listas

Una lista en una colección ordenada de objetos, que pueden tener diferente tipo, por ejemplo:

```
>>> l = [1, 2, 3, 4, 5]
>>> type(l)
<type 'list'>
```

Listas

accediendo a los elementos contenidos en una lista:

```
>>> l[2]
3
```

Contando desde el final con índices negativos:

```
>>> l[-1]
5
>>> l[-2]
4
```

Atención, el priimer elemento tiene índice 0

```
>>> l[0]
1
```

Listas

Slicing

```
>>> l
[1, 2, 3, 4, 5]
>>> l[2:4]
[3, 4]
```

Warning

Atención, la estructura **l[start:stop]** contiene los elementos con los índices i de manera que start $\leq$ i $<$ stop (i va desde start hasta stop-1). Por lo tanto , **l[start:stop] tiene (stop-start) elementos**.

Lists

Estructura del slicing: l[start:stop:step]

Todos los parámetros del slicing son opcionales:

```
>>> l
[1, 2, 3, 4, 5]
>>> l[3:]
[4, 5]
>>> l[:3]
[1, 2, 3]
>>> l[::2]
[1, 3, 5]
```

Listas

Los elementos de una lista pueden tener tipo diferente:

```
>>> l = [3, 2+3j, 'hello']
>>> l
[3, (2+3j), 'hello']
>>> l[1], l[2]
((2+3j), 'hello')
```

Listas

Python ofrece un amplio abanico de funciones para modificar listas o buscar en su interior. Aqui pondremos algunos ejemplos, para encontrarlas todas mirar en
http://docs.python.org/tutorial/datastructures.html#more-on-lists

Añadir y eliminar elementos

```
>>> l = [1, 2, 3, 4, 5]
>>> l.append(6)
>>> l
[1, 2, 3, 4, 5, 6]
>>> l.pop()
6
>>> l
[1, 2, 3, 4, 5]
>>> l.extend([6, 7]) # extend l, in-place
>>> l
[1, 2, 3, 4, 5, 6, 7]
>>> l = l[:-2]
>>> l
[1, 2, 3, 4, 5]
```

Listas

Invertir una lista

```
>>> r = l[::-1]
>>> r
[5, 4, 3, 2, 1]
```

Encadenar y repetir

```
>>> r + l
[5, 4, 3, 2, 1, 1, 2, 3, 4, 5]
>>> 2 * r
[5, 4, 3, 2, 1, 5, 4, 3, 2, 1]
```

Ordenar (al vuelo)

```
>>> r.sort()
>>> r
[1, 2, 3, 4, 5]
```

if/then/else

If

```
>>> if 2**2 == 4:
...     print 'Obviamente!'
...
Obviamente !
```

Los bloques que se ejecutan en las sentencias if se delimitan con tabuladores

```
a = 10
if a == 1:
    print(1)
elif a == 2:
    print(2)
else:
    print('muuuchos!')
```

```
muuuchos !
```

Evaluación de condiciones

El objeto if:

determina que una condición es falsa para...:

- Cuaquier nu´mero que sea cero (0, 0.0, 0+0j)
- cualquier contenedor vacio (lista, tupla, conjunto, diccionario, ...)
- False, None

determina que una condición es verdadera para...:

- cualquier otra cosa

Tests:

```
>>> 1==1.
True
```

Tests identidad

```
>>> 1 is 1.
False
>>> a = 1
>>> b = 1
>>> a is b
True
```

Comprobar si un nu´mero está en una lista

```
>>> b = [1, 2, 3]
>>> 2 in b
True
>>> 5 in b
False
```

EJERCICIOS BÁSICOS

Escriba scripts que resuelvan los siguientes problemas

- escriba un programa que determine si un polinomio cuadrático tiene cero una o dos raices reales. Pida al usuario los coeficientes del polinomio.

for/range

Iteraciones con un índice:

```
>>> for i in range(4):
...     print(i)
...
0
1
2
3
```

Pero es más comun iterar sobre elementos:

```
>>> for word in ('cool', 'powerful', 'readable'):
...     print('Python is %s' % word)
...
Python is cool
Python is powerful
Python is readable
```

while/break/continue¶

Un while típico en C (conjunto de Mandelbrot):

```
>>> z = 1 + 1j
>>> while abs(z) < 100:
...     z = z**2 + 1
...
```

Para romper una iteración for/while hay que usar break:

```
>>> z = 1 + 1j
>>> while abs(z) < 100:
...     if z.imag == 0:
...         break
...     z = z**2 + 1
```

Para continuar con la siguiente iteración hay que usar continue:

```
a = [1, 0, 2, 4]
for element in a:
    if element == 0:
        continue
    print 1. / element
```

```
1.0
0.5
0.25
```

Ejercicio

Problemas relacionados con while/for

- Escriba un programa que inicialmente tenga un nu´mero entre el 1 y el 10. El usuario debe acertar el nu´mero en 3 intentos, si lo acierta le felicitaremos, si no lo acierta nos reiremos de él.

Avanzado

Compute the decimals of π using the Wallis formula:

$$\pi = 2 \prod_{i=1}^{\infty} \frac{4i^2}{4i^2 - 1}$$

Avanzado.... muy avanzado

El muñeco de Mandelbrot

- $-2 < x < 1$, con 78 columnas, $-1.38 < y < 1.38$, con 36 filas y algoritmo $Z_{k+1} = Z_k^2 + C_0$

Si la trayectoria escapa ponemos " ", si la trayectoria no escapa ponemos "*"

Definición de funciones

Los bloques de código correspondientes a una función deben estar indentados, como el resto de bloques de control de flujo.

```
In [56]: def test():
    ....:     print('in test function')
    ....:
    ....:

In [57]: test()
in test function
```

El comando Return

Las funciones pueden retornar un valor si les hace falta, pero no es obligatorio.

```
In [6]: def disk_area(radius):
   ...:     return 3.14 * radius * radius
   ...:

In [8]: disk_area(1.5)
Out[8]: 7.0649999999999995
```

Estructura:

- La palabra clave def;
- seguida por el nombre que deseamos poner a la funcón, ... y
- los parámetros que enecesitará la función para hacer su trabajo entre corchetes cuadrados y separados por comas. DOS PUNTOS:
- el cuerpo de la función indentado
- y opcionalmente el retorno de los cálculos.
- Por defecto, si no retornamos nada python retorna None.

Parámetros

Imponer parámetros obligatorios(parámetros posicionales)

```
In [81]: def double_it(x):
   ....:     return x * 2
   ....:

In [82]: double_it(3)
Out[82]: 6

In [83]: double_it()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

/Users/cburns/src/scipy2009/scipy_2009_tutorial/source/<ipython console> in <module>()

TypeError: double_it() takes exactly 1 argument (0 given)
```

Parámetros

Imponer parámetros opcionales (parámetros con nombre)

```
In [84]: def double_it(x=2):
   ....:     return x * 2
   ....:
In [85]: double_it()
Out[85]: 4
In [86]: double_it(3)
Out[86]: 6
```

Ejercicio

Fibonacci

Escriba una función que calcule el elemento n de la serie de Fibonacci

Conversión de grados centígrados a farenheit

Escriba una función que convierta entre temperaturas escritas en grados farenheit y centigrados

Avanzado

Escriba una función que calcule el seno de un mu´mero utilizando la expansión de Taylor de la función seno