Introducción a Python

Andrés Ignacio Martínez Soto
anmarso4@fiv.upv.es

Seminarios Formación Polinux
València 20 Noviembre 2008

NULL: Licencia de ésta presentación

Usted es libre de:

* copiar, distribuir y comunicar públicamente la obra

* hacer obras derivadas

Bajo las condiciones siguientes:

Reconocimiento. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).

No comercial. No puede utilizar esta obra para fines comerciales.

Compartir bajo la misma licencia. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

* Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Contenido

Contenido

1. ¿Qué es Python?

1.1 ¿Qué es Python? (I)

- Lenguaje de programación
- Guido van Rossum principios años 90 cuyo
- Sintaxis muy limpia que favorece un código legible.

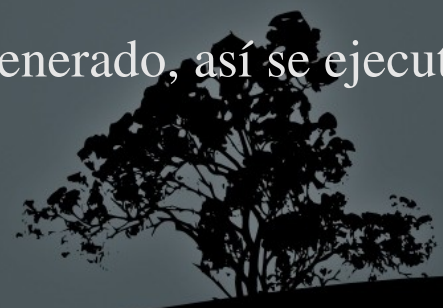
- Características
  - Lenguaje interpretado
  - Tipado dinámico y fuerte
  - Multiplataforma
  - Orientado a objetos

1.1 ¿Qué es Python? (I)

- Lenguaje interpretado:
  - Código fuente + intérprete de Python = Programa
  - Similar a Java:
    - El intérprete de Python ”compila” a bytecode el programa

    (lenguaje máquina optimizado para una máquina virtual de Python)

      - La 1ª vez que ejecutamos un programa éste compila a bytecode para la máquina virtual de Python.
      - Después se ejecuta el bytecode optimizado generado, así se ejecuta más rápido.



1.1 ¿Qué es Python? (II)

- Tipado dinámico
  - No es necesario declarar el tipo de dato que va a contener una determinada variable, éste se determina en tiempo de ejecución según el valor asignado
- Fuertemente tipado
  - No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita el tipo de la variable

Ejemplo:

```
>>> a="Cadena"; b=73; a+b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

1.1 ¿Qué es Python? (III)

- Multiplataforma:
  - Python es capaz de ser ejecutado en múltiples arquitecturas y sistemas operativos:
    - Arquitecturas: (”X86”,”ARM”,”Sparc” ...)
    - Sistemas Operativos: (”Windows”, ”GNU-Linux”, ”Unix”, ”Mac Os X”, ”Symbian” ...)

1.1 ¿Qué es Python? (IIII)

- Orientado a objetos:
  - La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa.
  - La ejecución del programa consiste en una serie de interacciones entre los objetos.
  - Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

1.2 ¿Por qué Python?

- Sintaxis clara, limpia y sencilla
- Tipado dinámico, gestión de memoria automáticamente
- Gran cantidad de librerías disponibles de serie
- Prototipado y desarrollo rápido de aplicaciones
- Lenguaje todoterreno

1.2 ¿Quién utiliza Python?

- Todas las distribuciones Linux
- Gnome, KDE
- Industrias Light & Magic
- Google, Yahoo
- Universitat Jaume I de Castelló en las carreras de Informática

1.3 Python VS otros

- Python,C y Java (¡Hola Mundo!)

Hola mundo en Python

```
print ”¡Hola Mundo!”
```

Hola mundo en C

```
#include <stdio.h>

int main (void) {
        printf(”¡Hola Mundo!”);
        return 0;
}
```

Hola mundo en Java

```
public class HolaMundo {
        public static void main(String [] args) {
                System.out.println(”¡Hola Mundo!”);
        }
}
```

1.4 Instalación de Python

- GNU-Linux:
  - Lo más probable es que ya venga instalado, de todas maneras:
    - sudo apt-get install python
- Windows:
  - Dirígete a http://www.python.org/download/ y bájate y ejecuta el instalador correspondiente
- Mac Os X:
  - Dirígete a http://www.python.org/download/ y bájate y ejecuta el instalador correspondiente

2. Introducción a Python

2. ¡Hola Mundo!

```
andreu@ilab:~$ python

Python 2.5.2 (r252:60911, Oct  5 2008, 19:24:49)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
>>> print "¡Hola Mundo! , soy un intérprete de Python"
¡Hola Mundo! , soy un intérprete de Python
```

2.1 Tipos de datos en Python (I)

- Python es un lenguage con tipado dinámico y fuerte...
- En Python existen 5 tipos predefinidos de variables;
  - Números: integers,floats,longs,complex ...
  - Cadenas: string
  - Vectores: tuplas,listas ...
  - Otros: diccionarios (hashes), objetos ...

2.1 Tipos de datos en Python (II)

- Numéricos:
  - Enteros:

```
>>> a=10; b=15;  c=a+b;  print "La suma de a+b es",c
La suma de a+b es 25
```

  - Reales:

```
>>> pi=3.14; f=2; r=5.50; long_circ=(f*pi*r);
print "La longitud de la circunferencia es ",long_circ
La longitud de la circunferencia es  34.54
```

  - Longs:

```
>>> a=9L; b=3L; a/b
3L
```

  - Complejos:

```
>>> a=1+0j; b=2+1j; a+b
(3+1j)
```

2.1 Tipos de datos en Python (III)

- Cadenas de carácteres:
  - Siempre comienzan por comillas simples ' , o comillas dobles ”

```
>>> init="Me gusta mucho"
>>> asignatura="MTP"
>>> centro="FIV"
>>> print "%s la asignatura %s de la %s" % (init,asignatura,centro)
Me gusta mucho la asignatura MTP de la FIV
>>> print init+" la asignatura "+asignatura+" de la "+centro
Me gusta mucho la asignatura MTP de la FIV
>>> cadena=init+" la asignatura "+asignatura+" de la "+centro
>>> print cadena
Me gusta mucho la asignatura MTP de la FIV
>>>
```

2.1 Tipos de datos en Python (IV)

- Estructuras de datos lineales:
  - Listas:
    - Puede haber listas de cualquier tipo de elementos (incluso de diferentes tipos), un elemento de una lista es mutable (se puede modificar)

```
>>> lista_cafeterias=["La Vella","Conservatorio","Trinquet","Àgora","Tony's"]
>>> for cafeteria in lista_cafeterias: print cafeteria
...
La Vella
Conservatorio
Trinquet
Àgora
Tony's
>>>
>>> otra_lista=[1,3+3j,"Cadenita"]
>>> lista_de_listas=[otra_lista,lista_cafeterias,[”MTP”,”TCO”,”SIO”,”ETC”] ]
```

    - Acceso a listas:

```
>>> lista_cafeterias[0]
'La Vella'
>>> lista_cafeterias[-1]
"Tony's"
>>> lista_cafeterias[:-1]
['La Vella', 'Conservatorio', 'Trinquet', 'Àgora']
>>>
```

2.1 Tipos de datos en Python (V)

- Tuplas:
  - No se pueden modificar los elementos de una tupla

```
>>> t = 12345, 54321, 'Foo'
>>> t
(12345, 54321, 'Foo')
>>> t[2]="Bar"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

2.1 Tipos de datos en Python (VI)

- Matrices:

```
import math
matriz2d=[ [1,2],[2,-12]]
mult=math.pi

for i in range(len(matriz2d)):
    for j in range(len(matriz2d[i])):
        matriz2d[i][j]*=mult

for i in range(len(matriz2d)):
    for j in range(len(matriz2d[i])):
        print matriz2d[i][j]

# Daría como resultado
3.14159265359
6.28318530718
6.28318530718
-37.6991118431
```

2.1 Tipos de datos en Python (VII)

- Otras estructuras de datos lineales:
  - Pilas:

```
>>> pila=[1,3,5,7]
>>> pila.pop()
7
>>> pila.append(7)
>>> pila[0]
1
>>> pila[-1]
7
>>>
```

  - Colas:

```
>>> queue = ["Eric", "John", "Michael"]
>>> queue.append("Terry")
>>> queue.append("Graham")
>>> queue.pop(0)
'Eric'
>>> queue.pop(0)
'John'
>>> queue
['Michael', 'Terry', 'Graham']
```

2.1 Tipos de datos en Python (VIII)

- Diccionarios
  - Hash, conjunto de pares ordenados {clave:valor}

```
asignaturas = {”mtp”:”dsic”,”etc2g”:”disca”}
>>>asignaturas[”mtp”]
dsic

asignaturas.has_key(”est”)
False

>>> asignaturas.keys()
['mtp', 'etc2g']

>>> asignaturas.items()
[('mtp', 'dsic'), ('etc2g', 'disca')]
```

2.2 Control de flujo (I)

- Condicionales:

```
If dia==”Jueves”:
        print ”Hoy hay charlas de Polinux”

elif dia==”Sábado”:
        print ”Hoy habrá fiesta”

else:
        print ”Hoy será un día aburrido”
```

  - Si se van a evaluar varias condiciones, mejor ponerlas entre paréntesis

```
If (dia>28) and (mes==2) and ( (anyo%100 !=0) or (anyo%400!=0):
        print ”Error, el anyo ”+anyo+ ”no es bisiesto”
elif (anyo%100==0) and (anyo%400==0):
        print ”El anyo ”+anyo+ ” es bisiesto”
else:
        pass
```

2.2 Control de flujo (II)

- Bucles:
  - Bucle for
    - Es un bucle foreach, diferente al de C/C++,Java ...
    - Está pensado para iterar sobre una secuencia de elementos

```
asignaturas=[”MTP”,”PRG”,”SIO”,”SOP”,”SOP2”,”RED”]
for asignatura in asignaturas:
        print ”Voy a aprobar ”+asignatura
```

    - Si queremos ”imitar” a la forma de C/C++ ... debemos hacerlo así:

```
asignaturas=[”MTP”,”PRG”,”SIO”,”SOP”,”SOP2”,”RED”]
for i in range(len(asignaturas)):
        print ”Voy a aprobar ”+asignatura[i]
```

```
for i in range(0,15000):
        if (EsPrimo(i)):
                print ”%d ¡es primo!”%(i)
```

2.2 Control de flujo (III)

- Bucles:
  - Bucle while:

```
while i<100:
        print i
        i=i*2


lista=range(0,10)
while len(lista)!=0:
        lista.pop()

while (valor<100) and (valor>200):
        valor=int(raw_input(”Dame un valor entre 100 y 200:  ”))
        if valor<100:
                print ”Necesito un valor mayor”
        elif valor>200:
                print ”Necesito un valor menor”
        else:
```

2.3 Entrada y salida (I)

- Entrada estándar:
  - 2 métodos:
    - raw_input(), y read() y sus derivados
    - raw_input() lee de stdin
    - read() lee de un flujo (fichero,stdin,socket ...)

```
nombre=raw_input("¿Cómo te llamas? ")
print "Encantado, "+nombre

import sys
...
print "Dame tu nombre: "
nombre=sys.stdin.readline()
print "Te llamas %s"%(nombre)
```

2.3 Entrada y salida (II)

- Salida estándar:
  - 2 métodos:
    - print() y write()
    - print() escribe en stdout
    - Write() escribe en un flujo (fichero,stdin,socket ...)

```
print ”¡Python mola!”

lang=”Perl”; caract=”ofuscado”
print ”¡%s es %s !” % (lang,caract)

import sys
sys.stdout.write(”Foo & Bar”)
```

2.4 Ficheros (I)

- Ficheros:
  - De estilo UNIX
  - Abrir fichero:
    - fich=open(nombre_fichero,modo)
    - Modos de apertura:
      - r → Lectura, texto
      - rb → Lectura, binario
      - w → Escritura, texto
      - wb → Escritura, binario
      - a → Añadir (append)
        - Si se le añade un + después del modo de apertura, si no existe el fichero, se crea.
  - Cerrar fichero:
    - fich.close()

2.4 Ficheros (II)

- Ficheros:
  - Leer:
    - fich.read(num_bytes) → lee num_bytes de un fichero , si no se especifica parámetro, se lee el fichero completo
    - fich.readline() → lee fich hasta encontrar '\n'
  - Ecribir:
    - fich.write(variable) → escribe variable en un fichero
    - fich.writeline(”Python”) → escribe Python\n en un fichero

2.4 Ficheros (III)

- Ejemplos:

```
fich=open(”fichero.txt”,”r”)
for linea in fichero:
    print linea
fich.close()
```

```
fich=open(”fichero.tx”,”r”)
linea=fich.readline()
print linea
while (True):
    linea=fich.readline()
    if (len(linea))==0:
        break

    print linea
fich.close()
```

```
try:
    fentrada=open(”entrada.txt”,”r”)
    fsalida=open(”salida.txt”,”w+”)

except:
    import sys
    print ”Ha fallado la apertura”
    sys.exit(-1)


while (True):
    linea=fentrada.readline()
    fsalida.write(linea)

fentrada.close()
fsalida.close()
```

2.4 Ficheros (IV)

- Serialización simple (cPickle):
  - Guarda y recupera cualquier tipo de datos en un fichero en formato binario.

```
>> import cPickle
>> datos = {'Nombre':'Andrés','Edad':22}
>> f=open(”datos.binarios”,”w”)
>> cPickle.dump(f,datos)
>> f.close()
>> file=open(”datos.binarios”,”rb”)
>> copia=cPiclke.load(file)
>> copia['Nombre']
   Andrés
```

2.5 Funciones (I)

- Se definen con la palabra reservada '*def*...

```
def nombre_función (parámetro1,*args,**kwargs):
    # Aquí el cuerpo de la función

def Saludar():
    Print ”!Hola Mundo!”

def SaludarconNombre (nombre):
    Print ”!Hola %s!” % (nombre)

def SaludarconNombre (nombre=”anónimo”):
    Print ”!Hola %s!” % (nombre)
```

nombre es el parámetro
que se le pasa a la función

nombre es el parámetro
que se le pasa a la función,
en este caso tiene un valor por defecto

2.5 Funciones (II)

- Llamadas a funciones:

```
def SaludarconNombre (nombre=”anónimo”):
      print ”¡Hola %s!” % (nombre)

>>> nombre=”Andrés”
>>> SaludarconNombre(nombre)
¡Hola Andrés!
```

SaludarconNombre es el nombre de la función en el espacio de nombres nombre es el parámetro pasado en la llamada a función

```
def Sumar(a,b):
      print a+b

>>> Sumar(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Sumar() takes exactly 2
arguments (1 given)
>>> Sumar(1,2)
3
```

Si no se le pasa el número correcto de argumentos, Python muestra una excepción de tipo TypeError

2.5 Funciones (III)

- El paso de parámetros en funciones se hace por referencia.
- Los cambios en variables sólo se 'reflejan' fuera de la función si el parámetro pasado es un objeto mutable (listas).
- Para devolver valores utilizaremos la palabra reservada '*return*'.
- La mejor forma de asegurarnos que la variable 'de fuera' se modifica tras ejecutar la función, es devolver la variable modificada.

```
def Incrementa(valor,cantidad):
    valor=valor+cantidad
    return valor

>>> val=10
>>> val=Incrementa(val,100)
>>> print val
110
>>>
```

2.5 Funciones (IV)

- Paso de parámetros *args y **kwargs:

```
def ImprimeLista (propietario,*lista):
      print ”Tengo una lista de ”+propietario
      for elemento in lista:
            print elemento
```


```
>>>ImprimeLista("Camarero","Bravas","Sepia")
Tengo una lista de Camarero
Bravas
Sepia
```

```
def varios(p1, **otros):
    print ”p1 vale ”+str(p1)
    for i in otros.items():
        print i

>>> varios("algo",uno="dos",dos=3)
p1 valealgo
('dos', 3)
('uno', 'dos')
>>>
```

2.6 Excepciones

- Una excepción se lanza cuando ha ocurrido algún error que Python no ”sabe” como manejar, y termina el programa:

```
def division(a,b):
    c=(a/b)
    return c

>>> division(10/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by
zero
```

La excepción, y su explicación

Excepción a capturar

```
try:
    division(10,0)
except ZeroDivisionError:
    print ”No se puede dividir por cero”
except:
    print ”Error inesperado”
finally:
    print division(10,1)
```

```
>>> try:
...     division(10,0)
... except ZeroDivisionError:
...     print "No se puede dividir por cero"
... except:
...     print "Error inesperado"
... finally:
...     print division(10,1)
...
No se puede dividir por cero
10
```

3. Módulos y paquetes

3.1 Módulos y paquetes

- Módulo: Conjunto de funciones, clases y variables guardadas en un fichero .py
- Paquete: Colección de módulos situados en un mismo directorio.
  - Para que un directorio sea reconocido como paquete es necesario que éste contenga un fichero de nombre __init__.py que puede estar vacío.
  - Un paquete puede contener subpaquetes (subdirectorios, cada uno con su __init__.py)

La diferencia entre
from paquete import * e import *
está en el espacio de nombres.

```
# paquete.py
def algo():
    Print ”algo”
```

```
>>> import math
>>> print math.cos(0)
1.0
```

```
import paquete
paquete.algo()
```

```
from paquete import * / algo:
algo()
```

```
# fichero funciones.py
def saludar (nombre):
    print ”Hola ”+nombre

def sumar (a,b):
    return (a+b)
```

```
#intérprete
>>> from funciones import *
>>> saludar(”Andrés”)
    Hola Andrés

>>> res=sumar(1+3)
>>> print res
    4
```

3.2 La librería estándar (I)

- Python tiene una amplia librería estándar implementada para casi todas las plataformas y sistemas operativos con muchísimos módulos útiles.
- Sistemas operativos:
  - sys y os
- Bases de datos:
  - Pysqlite3, MySQLdb
- Redes:
  - httplib,socket,urllib,xmlrpc,soap ...

3.2 La librería estándar (II)

- Hay muchos tipos de módulos para hacer casi cualquier cosa:
  - Procesamiento de texto, redes, bases de datos, sistemas operativos, cálculos complejos, gráficos, interfaces gráficas, páginas web, comunicaciones bluetooth …
- Consultar Python Standard Library:
  - http://effbot.org/librarybook/
- Consultar PyDoc:
  - http://pydoc.org

3.2 La librería estándar (III)

- Instalar un paquete es fácil:
  - Lo compilamos e instalamos nosotros mismos

```
cd paquete
python setup.py build
sudo python setup.py install
```

  - Python se encarga de hacerlo:

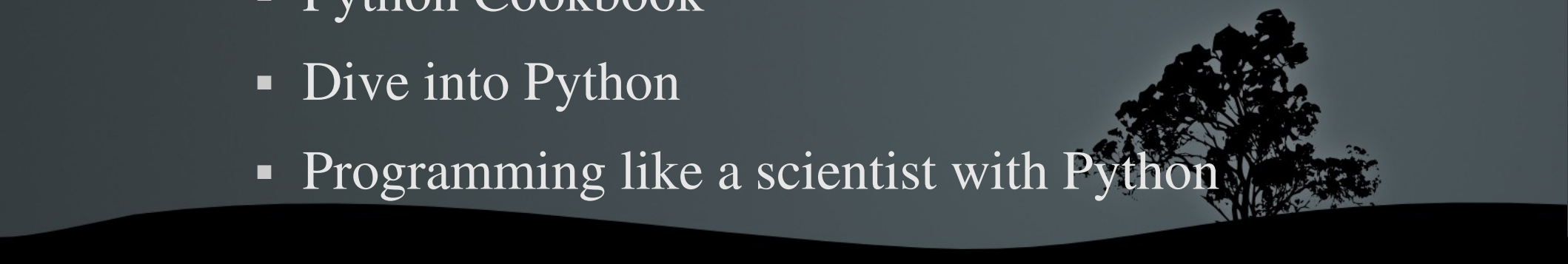
```
sudo easy_install nombre_paquete
```

4. Documentación

4.1 Libros sobre Python (I)

4.1 Libros sobre Python (II)

- Libros esenciales:
  - Programming Python
  - Learning Python
  - Python Standard Library
  - Introducción a la programación Python y C vol I
  - Python in a Nutshell
  - Python for Dummies
  - Python Cookbook
  - Dive into Python
  - Programming like a scientist with Python



4.2 Recursos online (I)

- Para iniciarse:
  - Introducción a la programación con Python y C volumen I, Python (Universitat Jaume I):
    - http://marmota.act.uji.es/ii04/teoria.shtml
  - Python para todos, Raúl González Duque:
    - http://mundogeek.net
  - Dive into Python:
    - http://diveintopython.org/
  - Charla ”Python avanzado” de Polinux:
    - http://www.polinux.upv.es

4.2 Recursos online (II)

- General:
  - Página web oficial de Python
    - http://www.python.org
  - Effbot (página web de Fredrik Lundh)
    - http://effbot.org
  - Lista python-es en Aditel
    - http://listas.aditel.org/archivos/python-es

- ¿Alguna pregunta o duda?

  ¡Gracias por vuestra atención!

- Andrés Ignacio Martínez Soto
- http://www.andresmartinezsoto.es
- anmarso4@fiv.upv.es

FIN

- END OF PRESENTATION