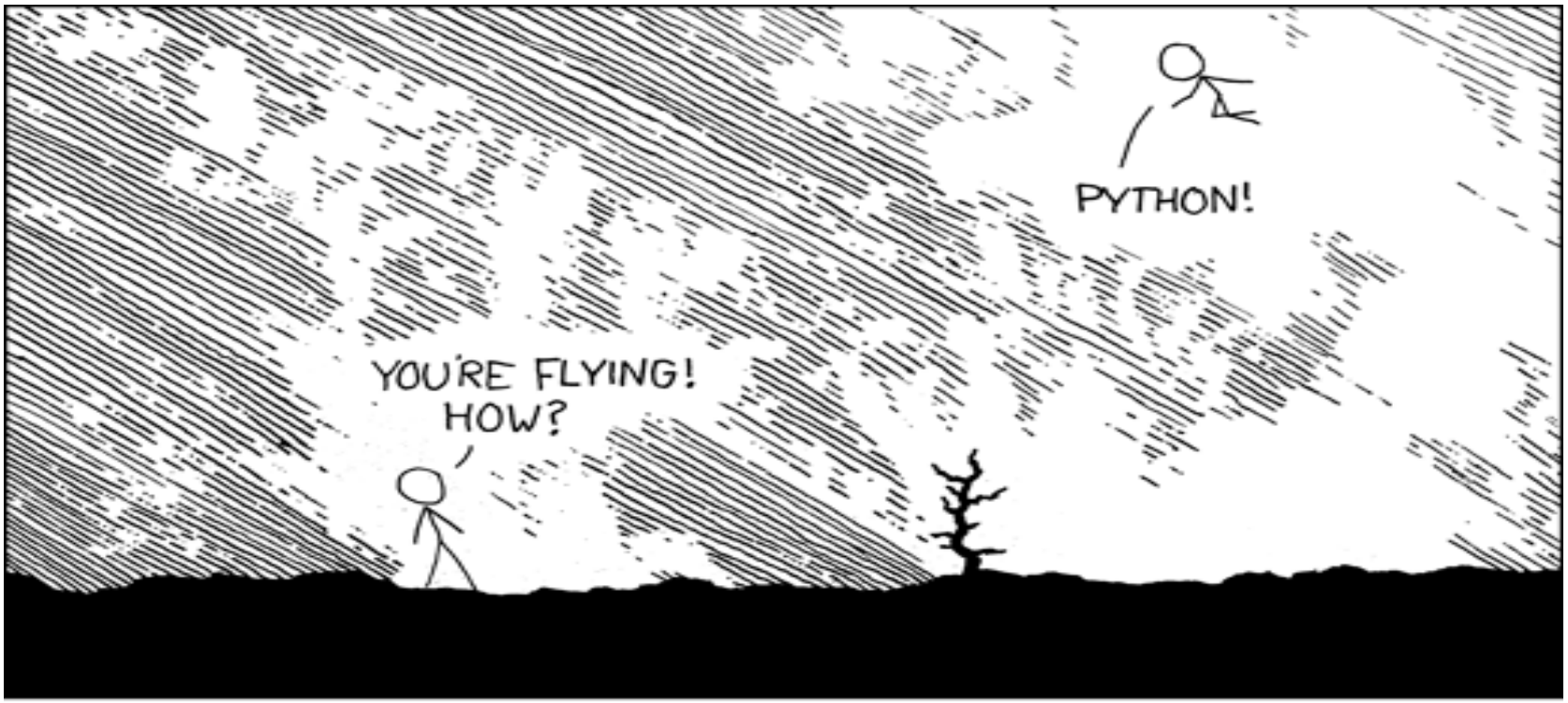
Fco Javier Lucena Lucena

Índice



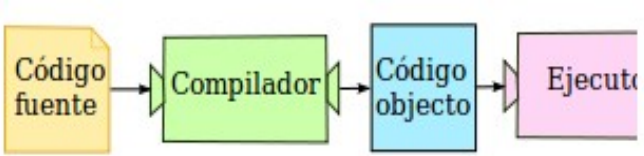
http://xkcd.com/

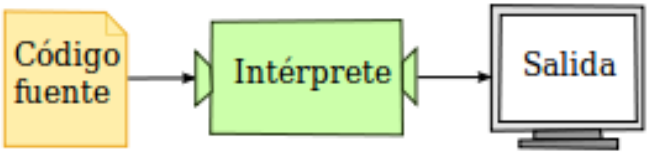
Introducción

- Python es un lenguaje de **programación dinámica** muy poderoso.
- Creado por Guido Van Rossum en 1991
- Comparado con Tcl, Perl, Ruby, Scheme o Java

Propiedades

- **Interpretado o de script**





  - Flexibilidad y Portabilidad
    - Pseudo código maquina intermedio (bitecode)
      - Ficheros .pyc

Propiedades

- **Tipado Dinámico**
  - No es necesario declarar el tipo de dato que va a contener una determinada variable.
  - El tipo se determinara en tiempo de ejecución.
  - El tipo puede cambiar si se asigna un valor diferente.

Propiedades

- **Fuertemente Tipado**
  - Python distingue “las churras de las merinas”
  - No se puede tratar una variable como si fuera de un tipo distinto al que tiene.
  - Ej: Sumar la cadena “1” y en entero 2

Why use it?

- Es un lenguaje “limpio”, tiene una sintaxis legible.
- Lenguaje muy parecido al pseudocodigo.
- Recomendado para empezar a programar.

Why use it?

- Python is **Open** !!
  - Usar
  - Distribuir
  - Incluso uso comercial
  - Python Software Foundation

  Licencia Python 2.6.2

Why use it?

- **Powerful and fast**
  - Se puede resolver un problema en cualquier ámbito.
  - La biblioteca estándar "lleva pilas incluidas"
    - Servidor web → 3 Lineas de código Python
  - Compilador de byte muy optimizado
    - *.pyc (bitecode)

Why use it?

- **Multiplataforma**
  - Windows, Linux/Unix, OS/2, Mac incluso Amiga
    .NET, Java... Dispositivos Móviles... Ej: Nokia
- Se **integra** con:
  - COM (extensiones windows)
  - bibliotecas Java (Jython),
  - .NET (IronPython)

- Fácil de **aprender**
  - Amplia y buena documentación en línea.

Aplicaciones

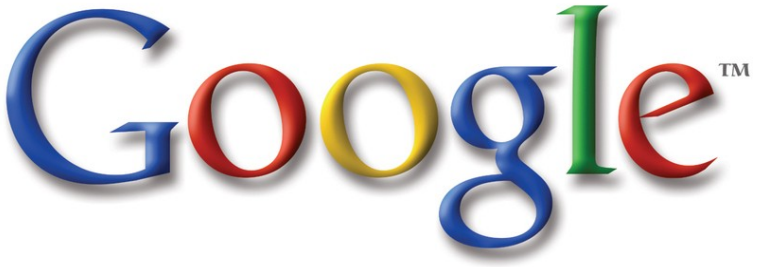
- Desarrollo Web e Internet
- Bases de Datos
- Desarrollo Software
- Juegos y Gráficos 3D
- Bioinformatica
- Física
- Educación

http://www.python.org/about/apps/

http://pypi.python.org/pypi

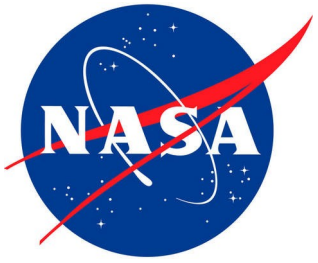
python powered

Casos de Éxito



INDUSTRIAL LIGHT & MAGIC

A LUCASFILM COMPANY



Instalación

- http://www.python.org/download/
- Disponible en la mayoría de distribuciones Gnu/Linux

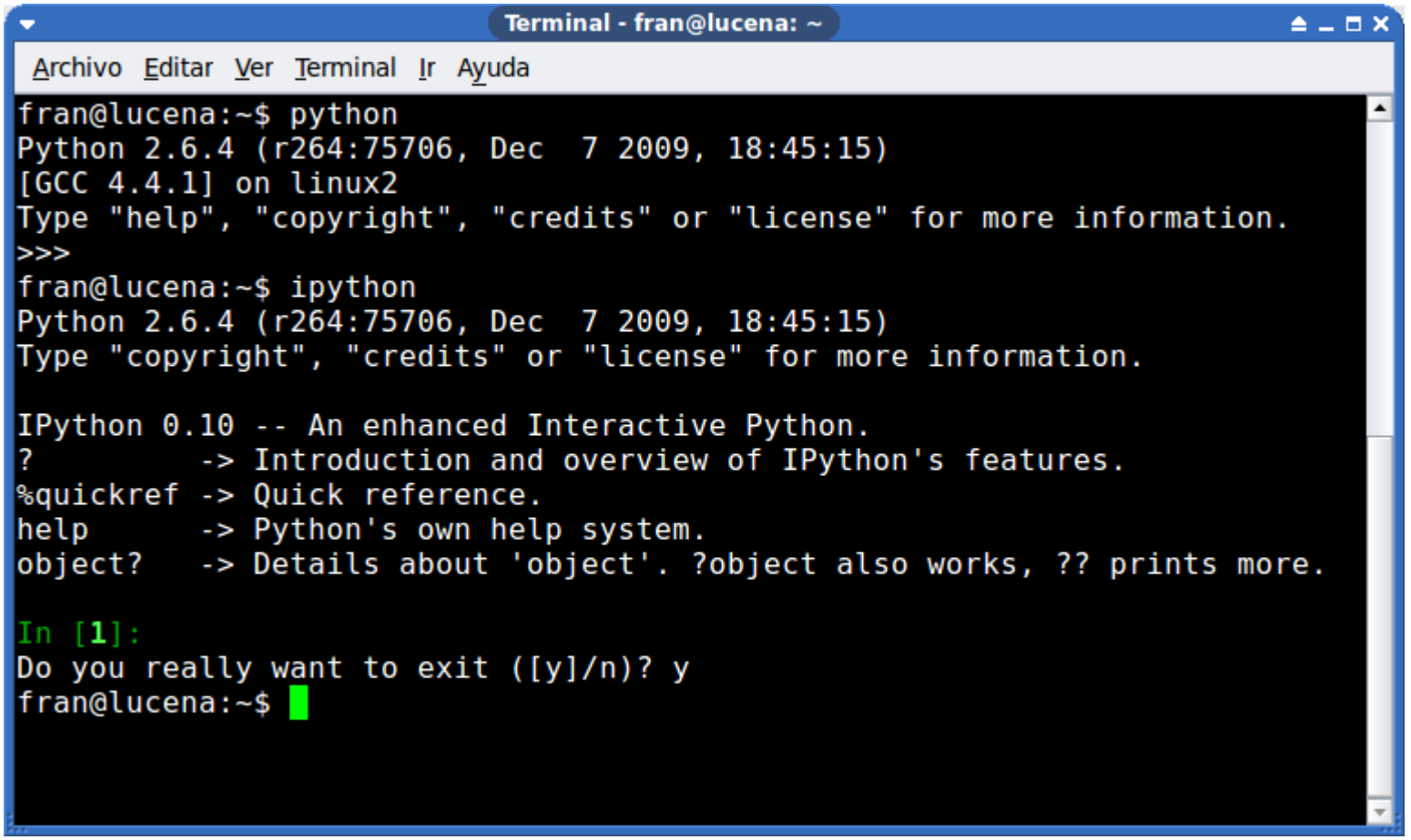
- sudo apt-get install **ipython**
  - http://ipython.scipy.org/moin/
  - Autocompletado (In[1] : pr + Tab)
  - Operador ? (Ayuda métodos)

- sudo apt-get install **geany**

Instalación

- Interprete Interactivo
  - Desde consola ejecutamos python o ipython

Control + d
Control + z



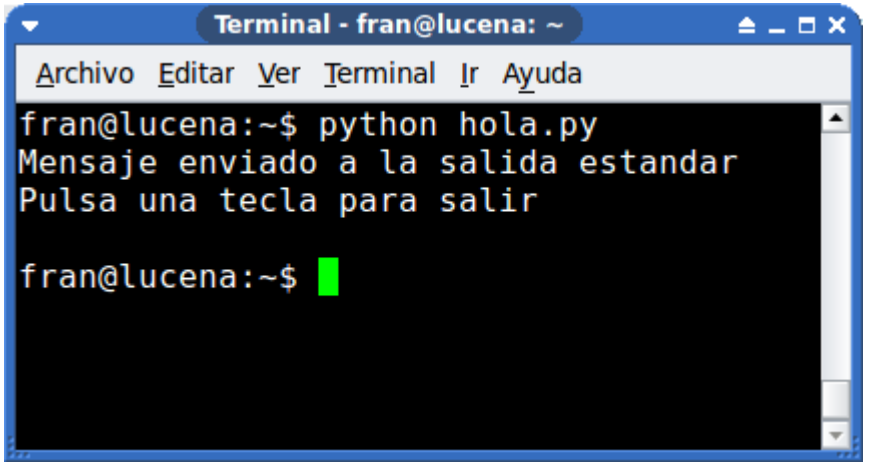
Instalación

- Script
  - Ejecutar un fichero desde la linea de comandos
  - Extensión ----> mi_fichero.py
  - Shebang ----> #!/usr/bin/python
  - python hola.py [./hola.py (chmod +x)]

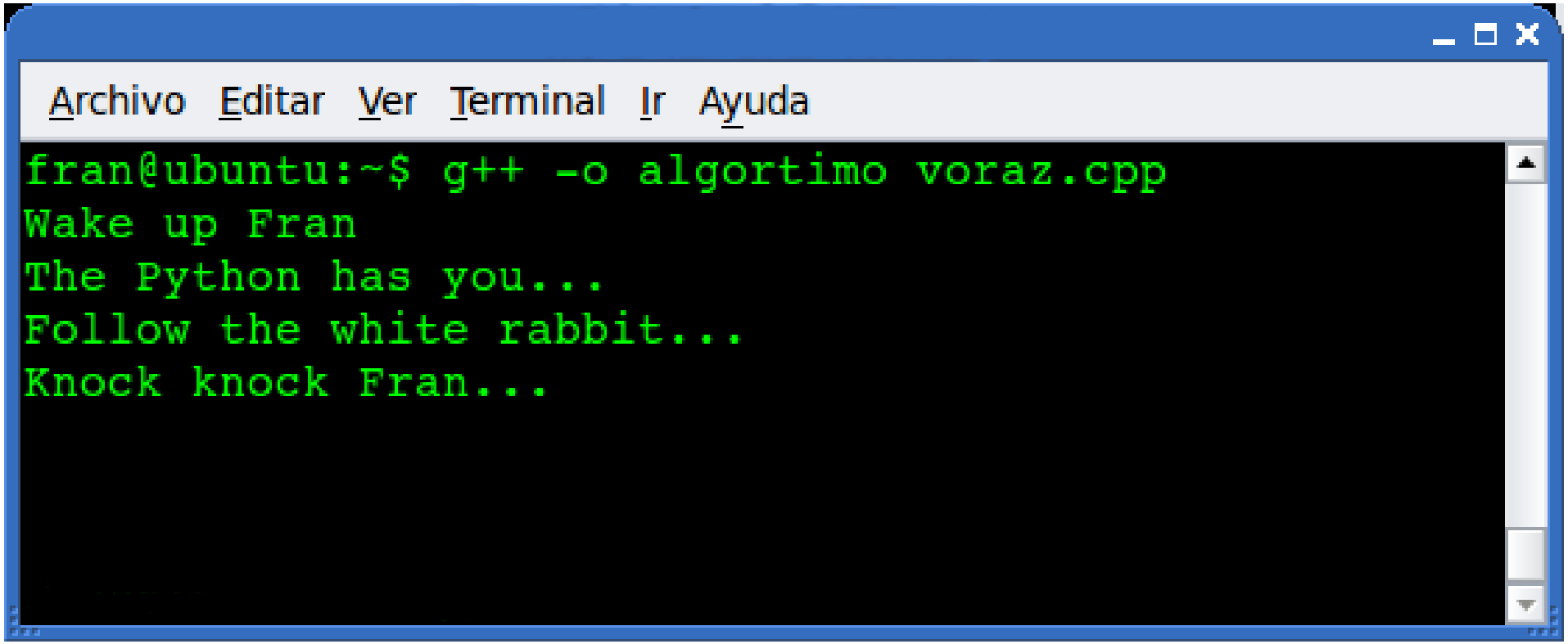
```
#!/usr/bin/python

print “Mensaje enviado”
print “Pulsa una tecla”

raw_input()
```



¿ Por qué empece a programar en Python ?



Sintaxis

- Se distinguen Mayúsculas – minúsculas
- No se necesita carácter para indicar el fin de una sentencia.
- Los bloques son especificados por indentación
- # Esta línea es un comentario
- Var = 3 # Asignación con el carácter =
- Var == Var1 # Operador igualdad
- Var -= 1 # Para incrementar Var += 1

Tipos de Datos

- Números : entero, flotante, complejo
- Cadenas de Texto
- Valores Booleanos
  - 3 , 4.57, 2 + 3j
  - “Hola Mundo”
  - True - False
- Saber el tipo de dato que contiene una variable: **type(id_variable)**

Variables

- No es necesario declarar las variables
  - Al utilizarlas se declaran automáticamente
- **Asignación Múltiple** en una línea
  - >>> a,b = 4,9
    >>> a
    4
    >>> a,b = b,a
    >>> a
    >>> 9

```
aux = a;
a = b;
b = aux;
```

Colecciones

- **Lista (vector – array)**
  - Puede contener cualquier tipo de dato
    - lista = [35, “Hola”, True, [1, 2, 3]]
  - Acceso a los elementos: lista[indice] indice=0,1…
    - lista[3][0] # Accede al elemento 1
  - [ ] Admite números negativos
    - lista[-2] devolvería el elemento True
  - lista[inicio:fin]
    - Lista[1:] #desde 1 a fin
    - Lsita [:2] # desde inicio a 2

Colecciones

- **Tupla**
  - Declaración: tupla = {1, True, “hola”}
  - Acceso: tupla[indice]
- Inmutables
  - Sus valores no se pueden modificar
  - Tamaño fijo
- Mas ligeras que las listas
  - Consumen menos memoria

Cadena de Texto

- cadena = “Que risa me da C”
- cadena[4] # devuelve r
- cadena[:3] # devuelve Que

Colecciones

- Diccionario
  - Relacionan una clave y un valor
- Acceso: dicc[clave]
- Dicc = {"75934029J" : "Lopez Lopez Jose,
  "64790681H" : "Perez Pepito}

Estructuras de Control

- Condicional

```
>>> x = int(raw_input("Introduce un número
entero: "))
>>> if x < 0:
...         x = 0
...         print 'Negativo cambiado a cero'
... elif x == 0:
...         print 'Cero'
... elif x == 1:
...         print 'Uno'
... else:
...         print 'Más'
...
```

switch
case

IndentationError: expected an indented block

Estructuras de Control

- Sentencia **for**
  - No hay valor inicial, ni condición de parada, ni un incremento.
  - Iterar por los elementos de una secuencia foreach en OO ó PHP

```
>>> # Medir algunas cadenas:
... a = ['gato', 'ventana', 'defenestrar']
>>> for x in a:
...     print x, len(x)
...
gato 4
ventana 7
defenestrar 11
```

Estructura de control

- Función **range()**

```
>>> a = ['Cinco', 'lobitos', 'tiene',
'la', 'loba']
>>> for i in range(len(a)):
...     print i, a[i]
...
0 Cinco
1 lobitos
2 tiene
3 la
4 loba
```

Estructuras de Control


```
numero = 23
detener = False

while not detener:
        adivina = int(raw_input('Un entero : '))

        if adivina == numero:
                print 'Enhorabuena, has acertado!.'
                detener = True # Esto causa que el
                               # bucle se detenga
        elif adivina < numero:
                print 'No, es un numero mayor.'
        else: # tienes que acertar
                print 'No, es un numero menor.'
else:
        print 'El bucle while ha terminado.'
        print 'Puedo hacer lo que quiero aqui.'

print 'Hecho.'
```

Funciones

```
>>> def fib(n):# escribir la serie Fibonacci hasta n
...
...     a, b = 0, 1
...     while b < n:
...         print b,
...         a, b = b, a+b
...
>>> # Y ahora llamamos a la función recién definida:
... fib(2000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
1597
```

Palabra reservada **return**

Funciones

- Documentar

```
def mi_funcion(param1, param2):
    """Esta función imprime los dos valores
            pasados como parámetros"""
    print param1
    print param2
```

Funciones

- Devolver Valores

```
def f(x, y):
    return x * 2, y * 2

a, b = f(1, 2)
```

Funciones

- Parámetros por defecto
- Parámetros Extra (tupla)

```
def varios(param1, param2, *otros):
    for val in otros:
        print otros

varios(1, 2)
varios(1, 2, 3)
varios(1, 2, 3, 4)
```

Funciones lambda

- Crear funciones anónimas en línea

```
def es_par(n):
    return (n % 2.0 == 0)
l = [1, 2, 3]
l2 = filter(es_par, l)
```

```
l = [1, 2, 3]
l2 = filter(lambda n: n % 2.0 == 0, l)
```

Módulos

- Fichero que contiene definiciones y sentencias.
- El nombre del fichero es el nombre del modulo sin la extensión .py
- Palabra reservada **import - from**
  - #!/usr/bin/python
    import nombre_modulo
    nombre_modulo.funcion()

    import nombre_modulo from mi_funcion

    mi_funcion()

    ....

Módulos

- Hay una gran variedad de módulos
- Antes de implementar, busca !
  - Probablemente esta ya hecho.
- Hay módulos matemáticos, web, programación evolutiva, inteligencia artificial

Módulos

- ¿Donde están?
- Variable de entorno: PYTHONPATH
  - >>> import sys
    >>> sys.path

Paquetes

- Sirven para organizar los módulos
- Para hacer que Python trate a un directorio como un paquete es necesario crear un archivo __init__.py
  - import paq.subpaq.modulo

    paq.subpaq.modulo.func()

http://pypi.python.org/pypi

Paquetes

Sonido/

- __init__.py
- Formatos/

  __init__.py

  leerwav.py

  escriwav.py

  leeraiff.py

  escriaiff.py

Clases

- Sintaxis mínima y nueva Semántica

```
class nombreClase:
    <sentencia-1>
    .
    .
    .
    <sentencia-N>
```

Clases

```
class MiClase:
    "Simple clase de ejemplo"
    i = 12345
    def f(self):
        return 'hola, mundo'
```

- MiClase, además de definir la clase, es un objeto en sí mismo.
- MiClase.i y MiClase.f son referencias a atributos válidas, que devuelven un entero y un objeto método, respectivamente

Clases

- La instanciación de clases utiliza notación de función.
  - x =ID_Clase()
- Esto crea una instancia de ID_Clase y la asigna a la variable x.
- Si queremos definir nuestro propio constructor, bastará con definir una función __init__

Clases

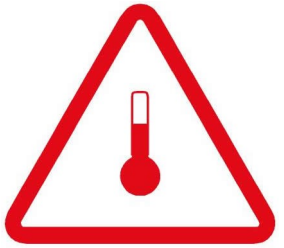
```
>>> class Complejo:
...     def __init__(self, parteReal,
parteImaginaria):
...         self.r = parteReal
...         self.i = parteImaginaria
...
>>> x = Complejo(3.0, -4.5)
>>> x.r, x.i
(3.0, -4.5)
```

Clases

```
class MiClase:
    "Simple clase de ejemplo"
    i = 12345
    def f(self):
        return 'hola, mundo'

X= MiClase()

x.contador= 1
while x.contador < 10
  x.contador = x.contador * 2
print x.contador
del x.contador
```



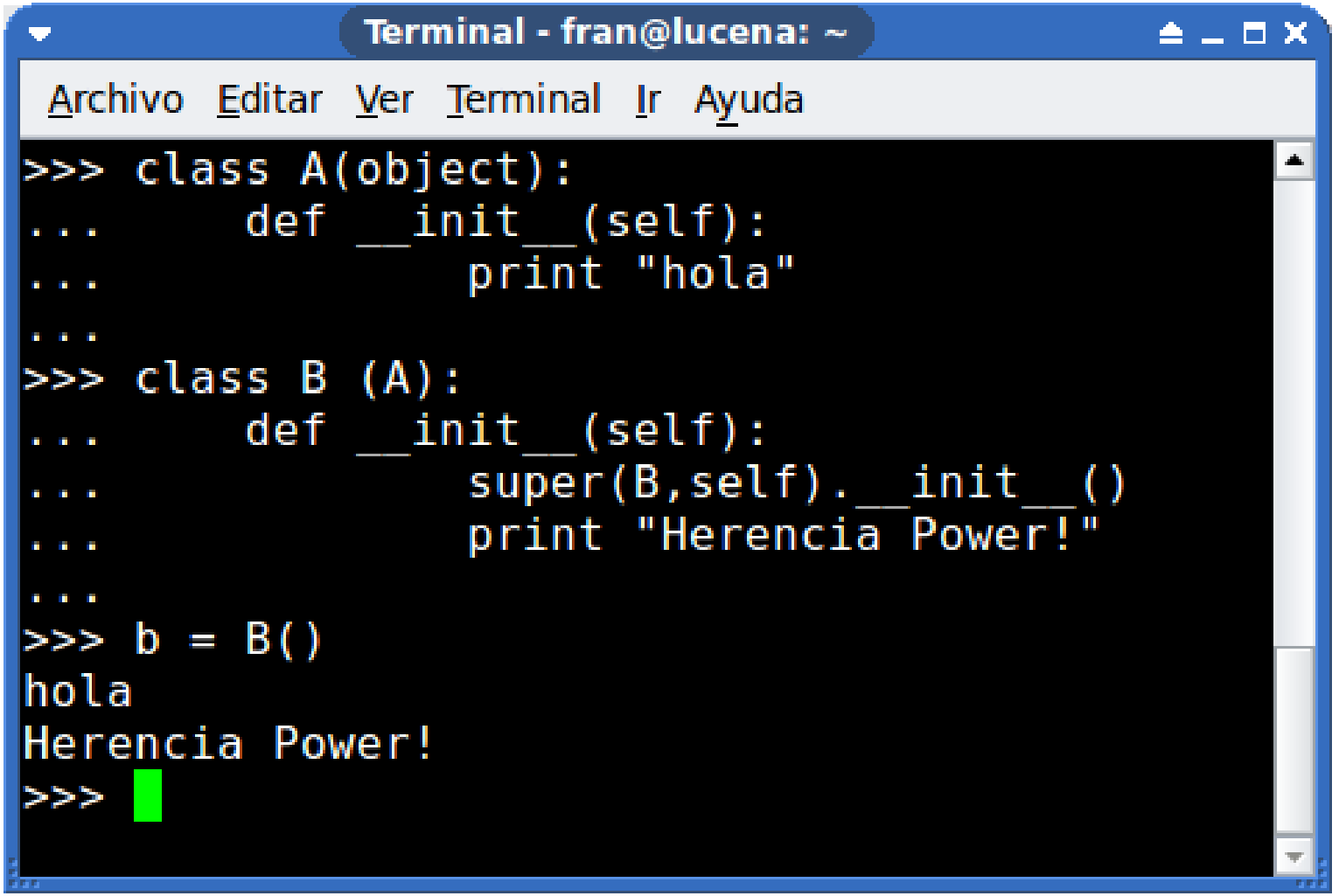
Clases

Método de clase
Método de instancia

```
class Bolsa:
    def __init__(self):
        self.datos = []
    def agregar(self, x):
        self.datos.append(x)
    def agregarDosVeces(self, x):
        self.agregar(x)
        self.agregar(x)
```

Clases

- Herencia



Biblioteca Estándar

- Sistema operativo

```
>>> import os
>>> dir(os)
<devuelve una lista de todas las funciones del
módulo>
>>> help(os)
<devuelve un extenso manual creado a partir de
las cadenas de documentación del módulo>

>>> os.getcwd()
/home/fran
```

Biblioteca Estándar

- Argumentos linea ordenes

python demo.py un dos tres

```
>>> import sys
>>> print sys.argv
['demo.py', 'un', 'dos', 'tres']
```

Biblioteca Estándar

- import re
- import math
- import urllib2
- import ramdom
- import datetime
- import threading
- import logging

Ejemplo

```
import urllib
import re

url = urllib.urlopen("http://eztv.it/frontpage.php")
html = url.read()
regexp = re.compile(r'<a href="([^"]+)"\s+class="download_1".+? \
<span class="epinfo">([^>]+?)</span>', re.I | re.MULTILINE | re.DOTALL)
torrents = regexp.findall(html)

for url,show in torrents:
    page = urllib.urlopen(url)
    fout = file("%s.torrent" % show, "wb+")
    try:
        print "Descargando '%s'" % show
        fout.write(page.read())
    finally:
        fout.close()
```

Ejemplo

```
import urllib2
import json

api_key = '----------------------'
method  = 'flickr.photos.search'
format = 'json'
tags = 'python'

url = 'http://api.flickr.com/services/rest/?method=%s \
&api_key=%s&format=%s&tags=%s&nojsoncallback=1' \
    % (method,api_key,format,tags)

file = urllib2.urlopen(url).read()

#print file

resultDictionary = json.loads(file)['photos']['photo']

print resultDictionary
```

Ejemplo

```
#yahoo_id:fr4nc1c
api_key = '--------------------'
method  = 'flickr.photos.search'
format = 'json'
nojsoncallback = '1'
tags = search_terms

url = 'http://api.flickr.com/services/rest/?method=%s&api_key=%s&format=%s& \
nojsoncallback=%s&tags=%s' \
% (method,api_key,format,nojsoncallback,tags)

file = urllib2.urlopen(url).read()

#creamos un diccionario con los datos que nos interesan
resultDictionary = json.loads(file)['photos']['photo']

resultList = []


for i in resultDictionary:
    AddResult(resultList, i)

#~ PrintFlickr(resultList)
return resultList;
```

python powered

Ejemplo

Ejemplo

Video Wii Motion + Python + Linux

Fuentes

- Python para todos
  - Raúl Gonzalez Duque
- Aprenda a Pensar Como un Programador con Python
  - Allen Downey Jeffrey Elkner Chris Meyers

Licencia

Fco Javier Lucena Lucena

fran.lucena@gmail.com

http://www.franlucena.es/