UNIVERSIDAD DE MÁLAGA

FGUMA FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

CURSOS ONLINE

FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

¿Por qué programar?

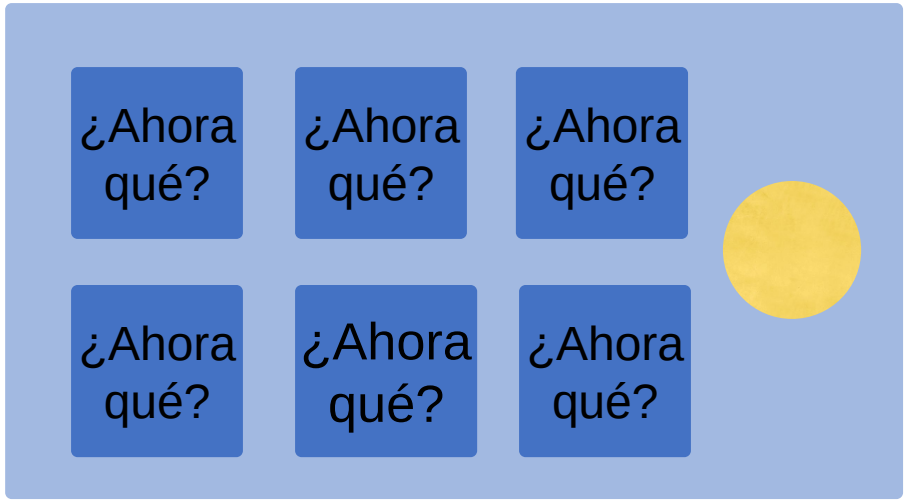
Unidad 1

Python para principiantes

dgm@uma.es

Los ordenadores están para ayudar...

- Están construidos con un propósito: hacer las cosas por nosotros
- Pero es necesario hablar su propio lenguaje para describir qué queremos que hagan
- Los usuarios lo tienen fácil: alguien le puso las instrucciones al ordenador en forma de programas, y ahora solo hay que elegir cuáles usar



Usuarios versus Programadores

- Los usuarios ven a los ordenadores como un conjunto de **herramientas**: procesador de textos, hoja de cálculo, mapa, lista de tareas, etc.
- Los programadores aprenden el "**comportamiento**" de los ordenadores y sus lenguajes
- Los programadores tienen herramientas que les permiten construir nuevas herramientas
- Los programadores a veces escriben herramientas para otros usuarios, y otras veces pequeñas "**utilidades**" para ellos mismos que les permiten **automatizar** tareas

¿Por qué ser un programador?

- Para realizar alguna tarea – somos el usuario y el programador
  - Limpiar los datos de la encuesta
- Para producir algo que otros usen – una tarea de programación
  - Solucionar un problema de rendimiento en el software Sakai
  - Añadir un libro de invitados a un sitio web

Desde el punto de vista del creador de software, **construimos** el software. Los usuarios finales son nuestros amos – a los que debemos agradar – y frecuentemente nos pagan cuando están contentos por nuestro trabajo. Nos enfrentamos a los **datos**, la **información** y las **redes** en lugar de los usuarios. El **hardware** y el **software** son nuestros amigos y aliados en esta búsqueda.

¿Qué es el Código? ¿El Software? ¿Un Programa?

- Una secuencia de instrucciones almacenadas
  - Es un producto de nuestra inteligencia en el ordenador
  - Se nos ocurre algo y lo codificamos, y se lo damos a alguien más para que **ahorre tiempo** cuando realice la misma tarea
- Una pieza de arte creativa – particularmente cuando hacemos que la experiencia del usuario sea buena

Programas para Humanos...

http://www.youtube.com/watch?v=vlzwuFkn88U

Programas para Humanos...

Mientras suena la música:

- Mano izquierda fuera y arriba
- Mano derecha fuera y arriba
- Girar mano izquierda
- Girar mano derecha
- Mano izquierda en el hombro derecho
- Mano derecha en el hombro izquierdo
- Mano izquierda en la cabeza
- Mano derecha en la cabeza
- Mano izquierda en el cinturón
- Mano derecha en el cinturón
- Mano izquierda en la nalga izquierda
- Mano derecha en la nalga derecha
- Contoneo
- Contoneo
- Salta

http://www.youtube.com/watch?v=vlzwuFkn88U

Programas para Python...

Averiguar la palabra que más aparece en un archivo

the clown ran after the car and the car ran into the tent and the tent fell down on the clown and the car

```
name = input('Nombre fichero:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

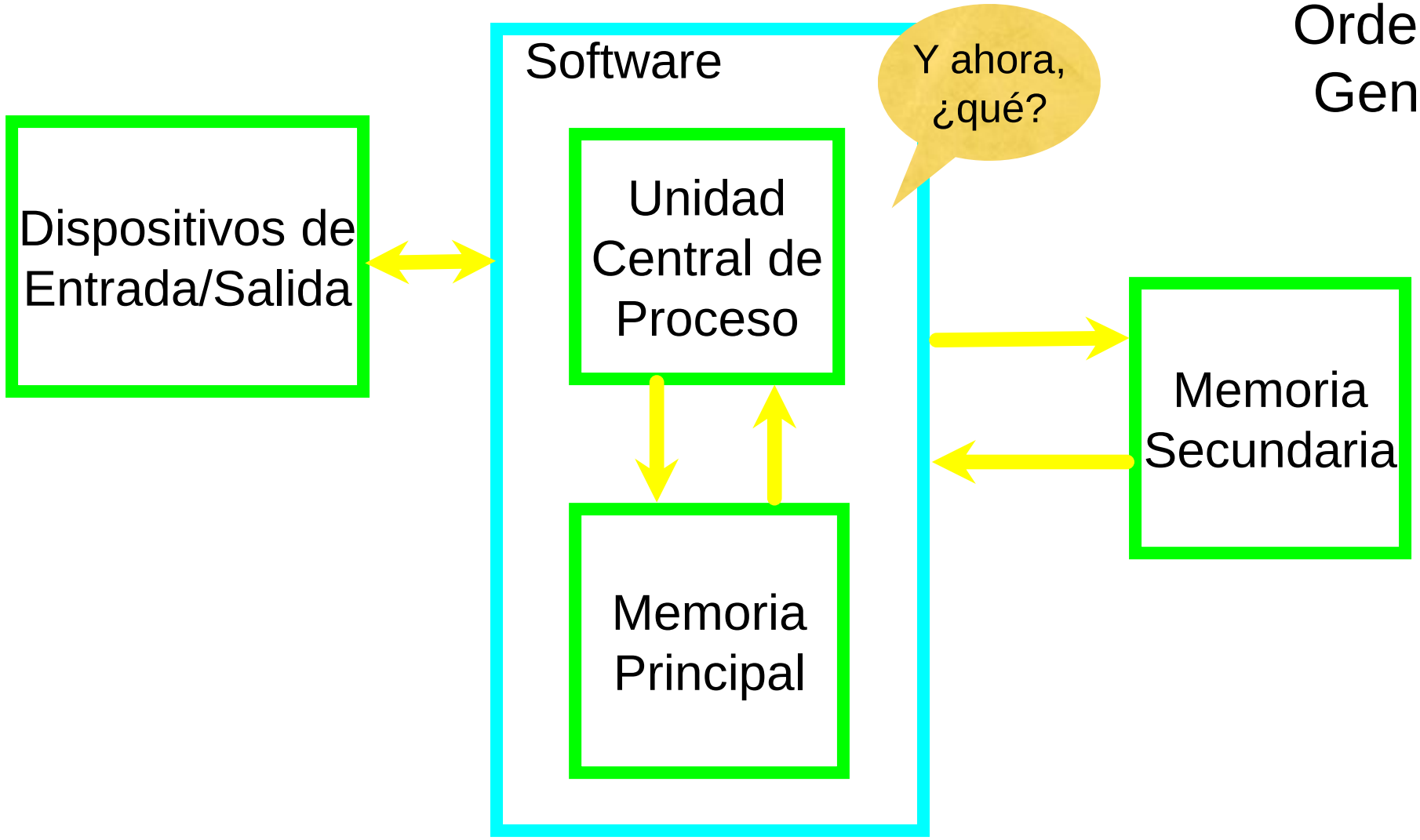
python words.py
Nombre fichero: words.txt
to 16

python words.py
Nombre fichero : clown.txt
the 7

Arquitectura Hardware

http://upload.wikimedia.org/wikipedia/commons/3/3d/RaspberryPi.jpg

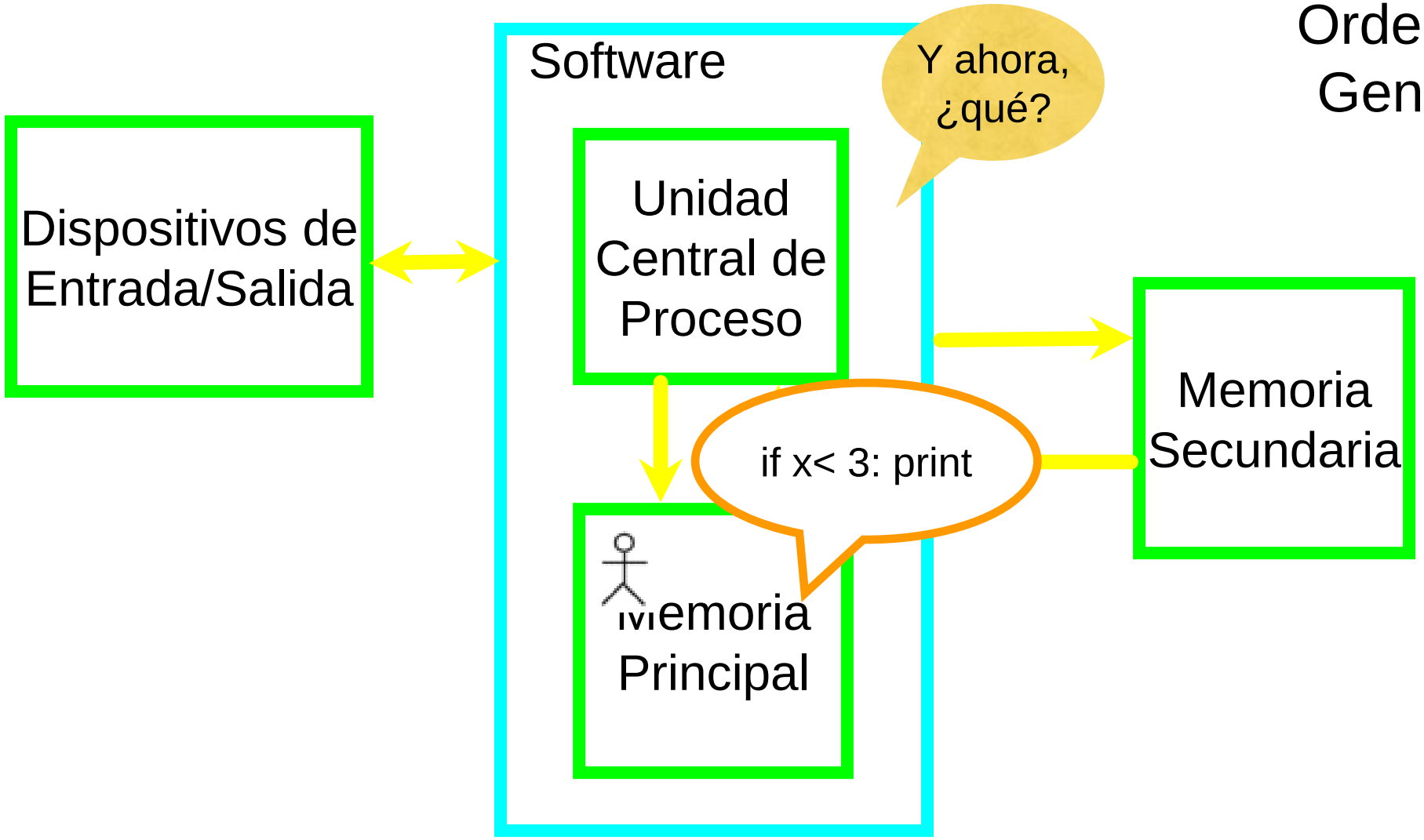
Ordenador Genérico

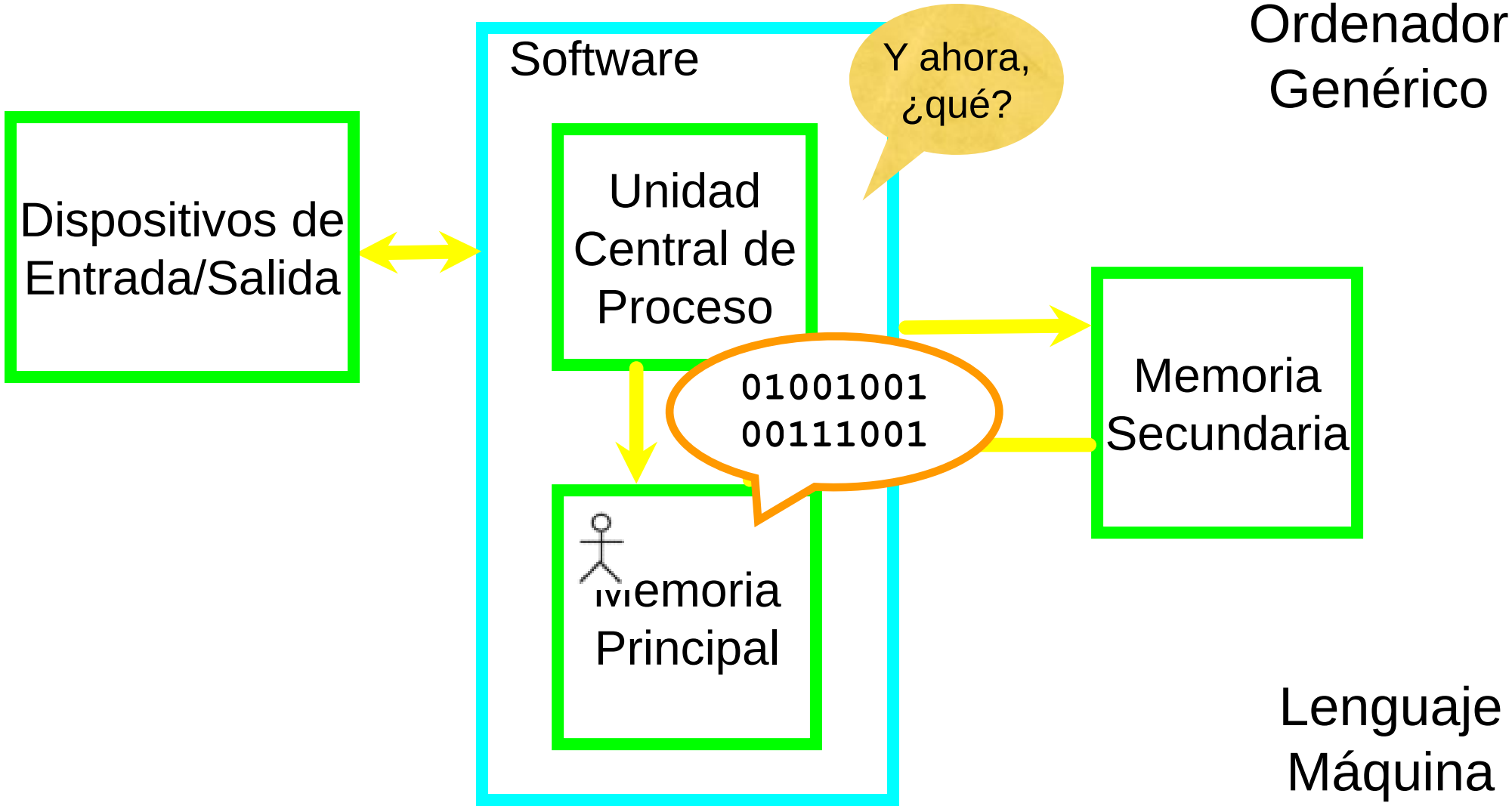


Definiciones

- **Unidad Central de Proceso (CPU)**: Ejecuta el programa - La CPU siempre está esperando "qué hacer a continuación". No exactamente un cerebro – muy torpe pero muy muy rápido
- **Dispositivos de Entrada**: Teclado, ratón, pantalla táctil
- **Dispositivos de Salida**: Pantalla, altavoces, impresora
- **Memoria Principal**: Almacenamiento temporal "pequeño" y rápido – se pierde tras un reinicio – memoria RAM
- **Memoria Secundaria**: Almacenamiento temporal más grande y lento – permanece hasta que se borre – disco duro / memoria USB

Ordenador Genérico




Ordenador
Genérico
Software
Y ahora,
¿qué?
Dispositivos de
Entrada/Salida
Unidad
Central de
Proceso
01001001
00111001
Memoria
Secundaria
Memoria
Principal
Lenguaje
Máquina

CPU quemada

http://www.youtube.com/watch?v=y39D4529FM4

Disco Duro en acción

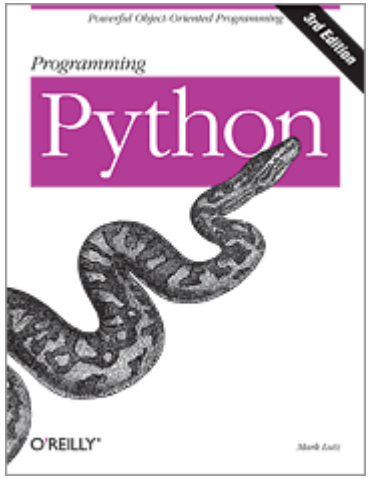
http://www.youtube.com/watch?v=9eMWG3fwiEU

Python como Lenguaje

Parsel es la lengua de las serpientes y de aquellos que pueden hablar con ellas. Alguien que puede hablar Parsel es conocido [en inglés] como un Parselmouth. Es una habilidad muy poco común, y es hereditaria. Prácticamente todos los hablantes de Parsel son descendientes de Salazar Slytherin.

http://harrypotter.wikia.com/wiki/Parseltongue

Python es el lenguaje del intérprete Python y de aquellos que pueden hablar con él. Alguien que puede hablar Python es conocido como un **Pythonista**. Es una habilidad muy poco común, y es hereditaria. Prácticamente todos los Pythonistas conocidos usan software desarrollado inicialmente por **Guido van Rossum.**



Principiantes: Errores de Sintaxis

- Necesitamos aprender el lenguaje Python para poder **comunicar** nuestras instrucciones a Python. Al principio cometeremos muchos errores y hablaremos farfullando como niños pequeños.
- Cuando cometes un error, el ordenador no tiene piedad. El mensaje que se obtiene es "**syntax error**" (error de sintaxis) – él conoce el lenguaje y tú lo estás aprendiendo. Parece que Python es cruel y sin sentimientos..
- Sólo recuerda que eres inteligente y que **puedes aprender**. El ordenador es simple y muy muy rápido, pero no puede aprender. Así que es más fácil para ti aprender Python que para el ordenador aprender español...

Hablando a Python

```
csev$ python3
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwinType
"help", "copyright", "credits" or "license" for more information.
>>>
```

Y ahora,
¿qué?

```
csev$ python3
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwinType
"help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

Esta es una buena prueba para asegurarte de que tienes Python correctamente instalado. Con **quit()** también puedes finalizar la sesión interactiva.

¿Qué le decimos?

Elementos de Python

- Vocabulario / Palabras - Variables y palabras reservadas (Unidad 2)
- Estructura de las sentencias – patrones de sintaxis válidos (Unidades 3-5)
- Estructura de programa – construyendo un programa para un propósito

```
name = input('Nombre fichero:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Una corta “historia” sobre cómo contar palabras con Python en un archivo

```
python words.py
Nombre fichero: words.txt
to 16
```

Palabras Reservadas

No se pueden utilizar las palabras reservadas como nombres / identificadores

```
False   class   return  is      finally
None    if      for     lambda  continue
True    def     from    while   nonlocal
and     del     global  not     with
as      elif    try     or      yield
assert  else    import  pass
break   except  in      raise
```

Sentencias o Líneas

```
x = 2
x = x + 2
print(x)
```

Sentencia de Asignación
Asignación con expresión
Sentencia print

Variable Operador Constante Función

Programando Párrafos

Scripts Python

- Interactuar con Python en el **intérprete** sirve para experimentos y programas de 3 o 4 líneas de largo
- La mayoría de los programas son mucho más largos, así que hay que **escribirlos en un archivo** y decirle a Python que ejecute las órdenes de ese archivo.
- En cierto sentido, estamos “dando a Python un guión” (**script**).
- Como convención, se suma **“.py”** como sufijo al final de estos archivos para indicar que contienen código Python.

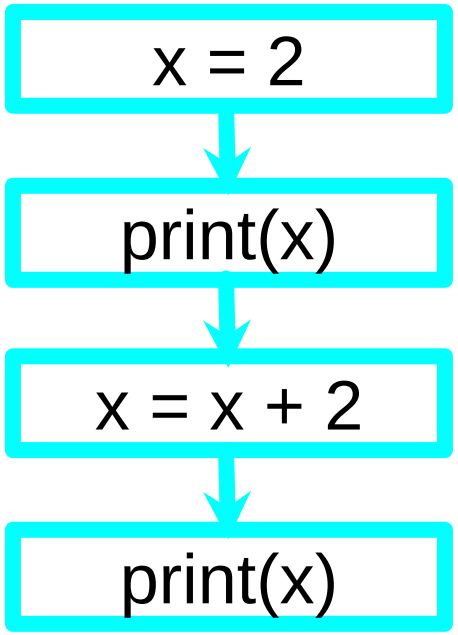
Interactivo versus Script

- Interactivo
  - Se escribe línea a línea cada vez y Python responde
- Script
  - Se introduce una secuencia de sentencias (líneas) en un archivo usando un editor de texto y Python ejecuta las sentencias del archivo.

Pasos de programa o Flujo de programa

- Como una receta o instrucciones de instalación, un programa es una secuencia de pasos para ser hechos en orden.
- Algunos pasos son condicionales – se pueden saltar.
- A veces un paso o grupo de ellos son repetidos.
- A veces almacenamos un conjunto de pasos para ser utilizados una y otra vez en diferentes partes de nuestro programa (Unidad 4).

Pasos Secuenciales



Programa:

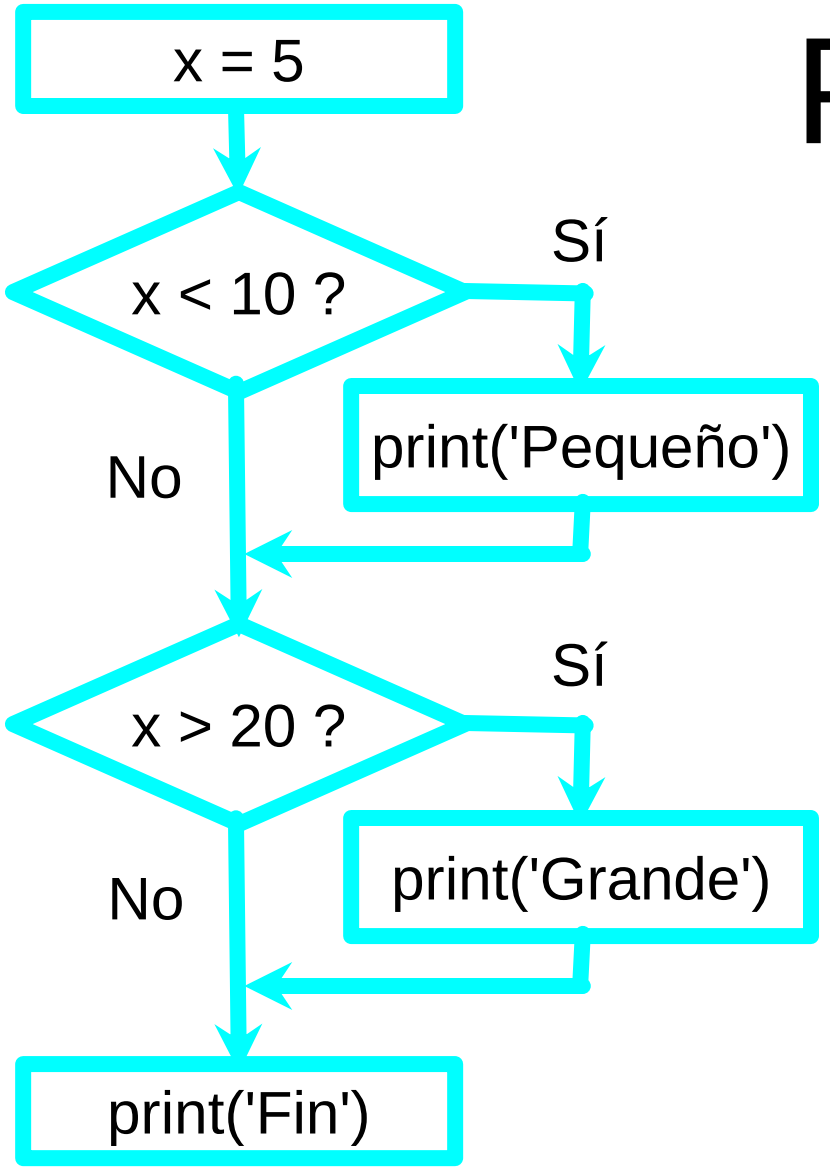
```
x = 2
print(x)
x = x + 2
print(x)
```

Salida:

2
4

Cuando un programa se está ejecutando, va de un paso al siguiente. Como programadores, establecemos “rutas” que el programa debe seguir.

Pasos Condicionales



Programa:

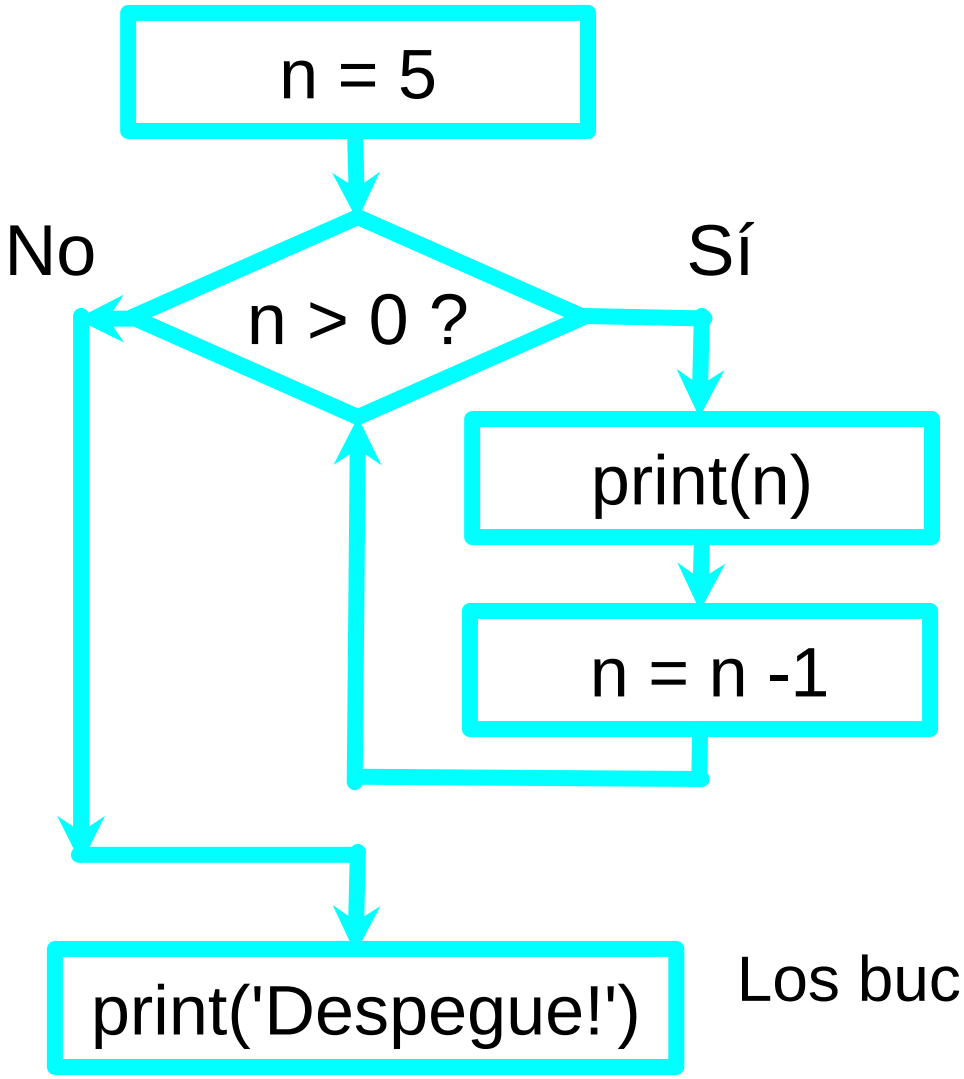
```
x = 5
if x < 10:
    print('Pequeño')
if x > 20:
    print('Grande')

print('Fin')
```

Salida:

```
Pequeño
Fin
```

Pasos Repetidos



Programa:

```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Despegue!')
```

Salida:

5
4
3
2
1
Despegue!

Los bucles (pasos repetidos) tienen variables de iteración que van cambiando a lo largo del bucle.

```
name = input('Nombre fichero:')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Secuencial

Repetido

Condicional

```
name = input('Nombre fichero:')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

Una “historia” corta en Python sobre cómo contar palabras en un archivo

Una instrucción utilizada para leer datos del usuario

Una sentencia sobre actualizar una de las muchas cuentas

Un párrafo sobre cómo encontrar el elemento más grande en una lista

Resumen

Esta es una visión rápida del contenido del curso

Volveremos a ver estos conceptos a lo largo del curso

Se centra en una visión global

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

… Insert new Contributors and Translators here
Spanish Version: Daniel Garrido (dgm@uma.es)

Continue…