

# Python

## Básico (II)

# print

- `>>> print "hola"`
- `hola`


 La *instrucción* **print** simplemente muestra en pantalla lo que le pasemos como argumento.

 En Python 3.x se escribe con otra sintaxis: **print("hola")**




# raw\_input()

- `>>> raw_input("Escribe algo: ")`
- `Escribe algo: |`

 La *función* **raw\_input()** espera a que introduzcamos algo y pulsemos 'intro'. Si nos interesa, podemos mostrar también un texto aclarativo.

# Variables

- `>>> hermanos = 3`
- `>>> hermanos - 1`
- `2`


 Una *variable* no es más que un contenedor que almacena algún tipo de valor (numérico, texto...). Allí donde se use, su valor lo sustituirá.

 ¡Usa nombres descriptivos para tus variables!



# #


- `>>> # Esto es un comentario que se ignora`
- `>>> print "Esto, sin embargo, se ejecuta"`
- `Esto, sin embargo, se ejecuta`

 Una *#* o *comentario* indica que lo que viene a continuación es información útil para el programador. Por su parte, Python, lo ignora en la ejecución.

 ¡Es importante que te organices!

# Codificación


● `-*- coding: utf-8 -*-`

 Para poder utilizar caracteres no anglosajones, como la ñ o los acentos, hemos de indicar la *codificación*. Lo más cómodo es usar la UTF-8 o *Unicode*.




# Autoejecución

- `#! /usr/bin/env python`

 Cuando hacemos doble click sobre un *archivo de texto ejecutable*, el sistema debe saber con qué lenguaje está escrito. Utiliza la directiva `#!` para indicárselo.

 ¡No olvides que **debe estar en la primera línea!**

# Módulos

 Un lenguaje de programación tiene unas funcionalidades básicas. Para extenderlas y hacerlo más potente, los programadores escriben librerías o *módulos*.


 Python viene con *pilas incluidas*.



# import


- `import random`
- `sorteo = random.randint(1,20)`

 Para incluir un módulo en un programa y poder usarlo se utiliza la instrucción **import**.

 Luego, puedes usar sus componentes utilizando la *notación dot*: **nombre\_modulo.nombre\_elemento**

# Bloques

```
● i = 1
● while i < 3:
●     print "i vale", i
●     i = i + 1
● # El bucle ha terminado.
```


 Los *bloques* se indican con : y todos sus contenidos están sangrados (habitualmente, 4 espacios).


 Una vez que el bucle termina, el sangrado desaparece.



# while

```
● i = 1  
● while i < 3:  
●     print "i vale", i  
●     i = i + 1
```


 El bucle **while** ejecuta su contenido una y otra vez mientras se verifique la condición indicada.

 En el ejemplo, se mostrarán en pantalla los valores 1 y 2 de la variable i.

# if ... elif ... else

```
• if 3 > 5:  
•     print "oro"  
• elif 3 = 5:  
•     print "plata"  
• else:  
•     print "bronce"
```

 El bucle **if** ejecuta su contenido si se cumple la condición indicada. Pueden usarse varias condiciones.


 En el ejemplo, se mostrará en pantalla el texto 'bronce'.



# break

```
• while True:  
•     print "i vale", i  
•     i = i + 1  
•     if i == 20:  
•         break
```

 La instrucción **break** fuerza la salida de un bucle.

 En el ejemplo, observa la acumulación de sangrados y el uso de los símbolos `==` y `=`.

# Tipos de datos



Números Enteros



Cadenas de Texto



Tuplas



Números Decimales



Booleanos



Listas



Diccionarios



 Números Enteros

● 27

● 1234567890L

 Números Decimales

● -27.36

 Cadenas de Texto

● 'Vaya toalla'

 Booleanos

● True

● False

 Listas

● [3 , 'calamar' , True]

 Tuplas


● (2.5 , 0 , -1e10)

 Diccionarios

● { 'Juan' : 3 , 'Pedro' : 5 , 'Ana' : 9 }

# conversiones


- `>>> str(3.2)`
- `'3.2'`
- `>>> list("hola")`
- `['h', 'o', 'l', 'a']`


 Unos tipos de datos pueden convertirse en otros usando funciones de Python (cuyos nombres son precisamente el del tipo de dato al que se quiere convertir).



# Objetos y Clases

- `>>> "hola".upper()`
- `'HOLA'`

 En realidad, en Python todo son **objetos**, entes que tienen sus propiedades y sus comportamientos.

 Además, puedes definir tus propias **clases** de objetos. O importarlas, como veremos, desde otros módulos.

¿Preguntas?