PYTHON

Programación básica

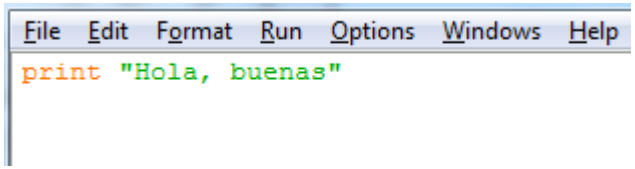
¿Qué es Python?

- Es un lenguaje de programación creado a principios de los 90
- Su nombre viene del grupo cómico Monty Python
- Es bastante sencillo de entender (comparado con otros lenguajes)
- Es multiplataforma (funciona tanto en Windows como en Linux y otros sistemas)
- Podemos escribir programas en Python utilizando distintos tipos de editores. Uno de los más utilizados es IDLE
- Una vez abierto Python, podemos escribir órdenes directamente en su consola. Por ejemplo, si escribimos *print "Hola, buenas"* sacará por pantalla el mensaje *Hola, buenas*

```
Python 2.5.1 (r251:54863, May 2 2007, 16:56:35)
[GCC 4.1.2 (Ubuntu 4.1.2-0ubuntu4)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Escribir y probar programas Python

- Lo normal no es ir escribiendo las órdenes una a una y ejecutándolas como en el ejemplo anterior, sino tener un fichero con un conjunto de órdenes o comandos de Python y ejecutarlo todo de golpe
- Para ello, desde IDLE, debemos ir al menú **File > New Window**. Se abrirá una nueva ventana que deberemos guardar como un fichero con extensión *.py* (por ejemplo, *Hola.py*).
- Una vez tengamos el fichero listo y guardado, podemos probarlo desde el menú **Run > Run Module** (o pulsando **F5**), desde la misma ventana donde estamos escribiendo el código



Ejercicio

- Crea un archivo en Python llamado **Tarjeta.py**. Rellena el código con instrucciones *print* como la del ejemplo (la puedes repetir tantas veces como quieras) para mostrar una tarjeta de visita con tus datos, algo parecido a esto:

```
==============================================
|             Juan García Peñalver           |
|          Estudiante de 1º de BACH A        |
|             IES número 23 (Almansa)        |
==============================================
```

Pedirle datos al usuario

- Escribir un programa que sólo saque datos por pantalla sin pedirle nada al usuario no es habitual. El usuario debe interactuar con el programa
- Para pedirle datos al usuario, se usa la instrucción **input()** (si queremos que el usuario introduzca un número entero) o **raw_input()** (si queremos que introduzca cualquier otra cosa)
- Modifica el proyecto **Hola.py** que has creado al principio, y añádele estas líneas (las que aparecen subrayadas)

```
print "Hola, buenas"
raw_input()
print "Fin"
```

- Vuelve a ejecutar el programa (**F5**)
- Observa que el mensaje segundo ("Fin") no aparece hasta que tú no escribas algo y pulses *Intro*.

Ejercicio

- Crea un proyecto llamado **Saludo.py**. Rellénalo usando las instrucciones que has visto hasta ahora para que el programa le pregunte al usuario su nombre, y luego le salude con ese nombre. Algo parecido a esto:

```
Hola, dime tu nombre:
Juan
Hola Juan
```

NOTA: la segunda línea la escribiría el usuario (utiliza *raw_input*()), la primera y tercera las saca el programa (utiliza *print*)

Comentarios en el código

- A veces un programa es algo largo o confuso, y viene bien tener algunas notas aclaratorias entre el código que sirvan para:
  - Separar unas zonas de código de otras
  - Aclarar qué hace una parte del código
- Estas anotaciones se llaman **comentarios**, y no son instrucciones, sino un texto cualquiera que luego no se compila, y nos sirve para entender mejor el código
- Los comentarios en Python se ponen con una almohadilla # delante del texto del comentario, en la misma línea

```
# Saludamos al usuario
print "Hola, buenos días"  #Ya hemos saludado
```

¿Qué hacemos con los datos del usuario?

- Pedirle datos al usuario está muy bien, pero ¿qué hacemos con ellos? ¿cómo podemos utilizarlos?
- Necesitamos una forma de guardarnos lo que ha introducido el usuario para poderlo utilizar. Ese mecanismo se llaman **variables**
- Una variable es una especie de “recipiente” donde guardamos un dato para utilizarlo cuando queramos
- Modifica el proyecto *Hola.py* del inicio, y añade el texto que aparece subrayado a continuación

```
print "Hola, buenas"
valor = raw_input()
print "Has escrito", valor
print "Fin"
```

Ejercicios

- Vuelve al proyecto **Saludo.py** que habíamos creado antes para saludar al usuario por su nombre. Ahora sí podemos completarlo correctamente con lo que sabemos.
  - Guárdate el nombre del usuario en una variable, y úsala para luego sacar ese nombre por pantalla en el saludo.
- Cuando termines, crea otro proyecto llamado **SumaSimple.py**. Vamos a hacer un programa que le pida al usuario dos números y luego los sume. Rellena el código con algo como esto:

```
print "Dime un numero:"
num1 = raw_input()
print "Dime otro numero:"
num2 = raw_input()
print "Resultado:"
print num1+num2
```

¿Por qué hemos creado dos variables diferentes llamadas *num1* y *num2*?

Prueba a ejecutarlo cuando termines. ¿Qué problema ves?

Tipos de datos

- Lo que le pedimos al usuario con *raw_input()* se trata como un texto simple
- A veces, lo que el usuario introduce es algo más que un texto. Puede ser una fecha, o un número, u otra cosa
- En general, casi cualquier lenguaje de programación admite trabajar con distintos tipos de datos simples:
  - **Alfanumérico o cadena**: para representar un texto, o conjunto de caracteres
    - **Ejemplo**: el nombre de una persona es una cadena
  - **Numérico**: para representar valores numéricos, que pueden ser **enteros** o **reales**
    - **Ejemplo**: la edad de una persona es numérico entero. El precio de un videojuego es numérico real (puede tener decimales)
  - **Booleano o lógico:** para representar valores de **verdadero** o **falso.**
    - **No son habituales en la vida cotidiana, pero sí en programación**
    - **Ejemplo**: ver si un número es mayor que cero, ver si dos datos son iguales

Tipos de datos: ejercicio

- Indica el tipo de dato más adecuado para almacenar la siguiente información
  - Un DNI sin letra
  - Un DNI con letra
  - El nombre de un departamento
  - El sexo de una persona (M/F)
  - El peso en Kg de una persona (con decimales)
  - El año de nacimiento de una persona
  - Indicar si una persona está casada o no
  - El resultado en la quiniela de uno de los partidos
  - El valor de uno de los números de la primitiva

Tipos de datos y variables en Python

- En Python existen los tres tipos de datos básicos vistos antes (cadenas, números y booleanos)
- Podemos crear variables automáticamente de cada uno de estos tipos de datos y darles valor. Por ejemplo:

```
numero = 32
numero2 = 2.25 + numero
casado = True
nombre = "Pepe"
```

- Observa que para los valores booleanos se pone *True* o *False* (empezando por mayúsculas)
- Pero... ¿qué pasa si queremos que el usuario introduzca el valor de la variable por teclado? Si es numérico usamos *input()*, y si no usamos *raw_input()*

```
numero = input()
nombre = raw_input()
```

Ejercicio

- Haz correctamente el programa **SumaSimple.py** que habíamos dejado incorrecto en páginas anteriores. Debe pedirle al usuario dos números enteros, y mostrar el resultado de la suma.

Operaciones aritméticas en Python

- Con los datos y variables podemos hacer varias operaciones aritméticas en Python. Para cada una de ellas usamos un **operador**.
- Los operadores aritméticos más habituales son
  - Suma: **+**
  - Resta: **-**
  - Multiplicación**: ***
  - División (con decimales): ***/***
  - División (entera): ***//***
  - Resto de división entera: **%**
  - Potencia: ****
- Si hacemos una operación con dos números enteros, el resultado será ENTERO. Si alguno de los datos es real, el resultado será REAL

```
num1 = 2
num2 = 3
num3 = 2.5
resultado1 = num1 + num2    # Daría 5
resultado2 = num2 ** num1   # Daría 9
resultado3 = num2 % num1    # Daría 1
resultado4 = num3 * num2    # Daría 7.5
```

Operaciones de comparación en Python

- Sirven para comparar valores, para ver si son iguales o diferentes, o ver cuál es mayor o menor
- Los operadores son > (mayor que), < (menor que), >= (mayor o igual), <= (menor o igual), == (iguales) y != (distintos)
- Dan como resultado un valor de True o False

```
num1 = 2
num2 = 4
num1 > num2    # FALSO
num1 < num2    # VERDADERO
num1 == num2   # FALSO
num1 != num2   # VERDADERO
```

Precedencia de operadores en Python

- Si tenemos varios operadores en una misma expresión, ¿cuál se hace antes?
  - **Ejemplo:** la expresión 4+2*6 vale diferente según si hacemos primero la suma (entonces daría 36) o si hacemos antes la multiplicación (entonces daría 16)
- Existen unas normas a la hora de evaluar los operadores de una expresión, de forma que se evalúan en este orden:
  - Primero los paréntesis que pongamos **(...)**
  - Después las potencias (**)
  - Después multiplicaciones, divisiones y restos
  - Después sumas y restas
  - Después comparaciones
  - Después asignaciones
- Ejemplos

```
(4+2)*6     # Daría 36
4+2<5       # Daría FALSO (6 < 5 es FALSO)
```

Alguna cosa más sobre la salida de datos

- Hemos visto que **print** sirve para sacar datos al usuario
- Podemos usar la coma para enlazar varias cosas a mostrar por pantalla

```
print "Hola", 4  # Daría Hola 4
```

- Podemos usar algunos símbolos especiales para hacer algunos efectos: **\n** para pasar a la siguiente línea, o \t para hacer una tabulación hacia la derecha

```
print "Hola\nBuenas"
```

- Daría como resultado:

```
Hola
Buenas
```

- Podemos usar el operador + para enlazar textos, o el operador * para repetirlo tantas veces como se quiera:

```
dato1 = "Hola"
dato2 = dato1 + dato1 # Daría HolaHola
dato3 = dato1 * 3     # Daría HolaHolaHola
```

Alguna cosa más sobre la salida de datos (II)

- El comando **print** también admite que, entre el texto, se pongan códigos para dar formato a algunos datos.
- Básicamente consiste en utilizar el símbolo **%** seguido de las letras **s**, **d** o **f** según si lo que queremos formatear es un texto, un número entero o un número real
- En el caso de números reales, podemos especificar con números delante de la f cuántas cifras decimales queremos mostrar.

```
nombre = "Pepe"
edad = 24
peso = 78.86
print "Hola %s" % (nombre) # o bien print "Hola", nombre
print "Tienes %5d años y pesas %.1f kg" % (edad, peso)
#Daría: Tienes    24 años y pesas 78.9 kg
```

Ejercicio

- Rehaz el ejercicio del proyecto **Tarjeta.py** hecho antes, para sacar tu tarjeta de visita utilizando una sola instrucción *print*