

Paradigmas de Programación

Sesión 1 Laboratorio



Cristian Tejedor García
Departamento de Informática
Universidad de Valladolid

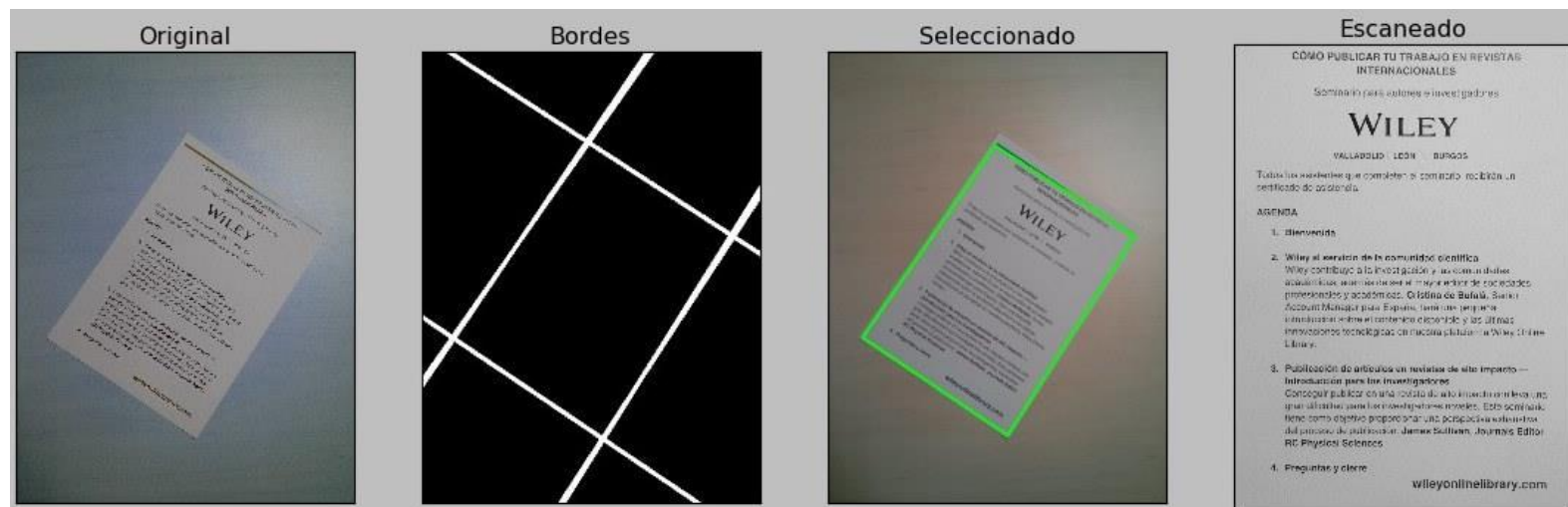
Curso 2016-17

Grado en Ingeniería Informática
INDAT

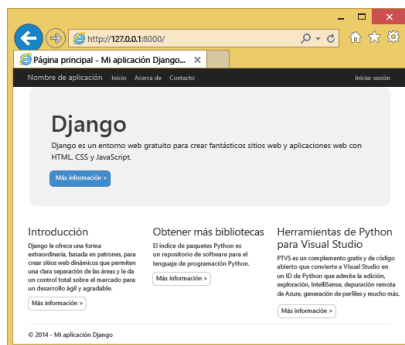


0. Motivación Python

Potente procesamiento de imágenes (OpenCV)



Web (Django)



Escritorio (juegos)



Scripting





Contenido

0. Motivación Python
1. Objetivos
2. Horario
3. Entrega de prácticas
4. El lenguaje Python
5. Sesiones de laboratorio
6. Mi primer programa en Python
7. Tipos básicos
8. Colecciones
9. Entrada/Salida
10. Ficheros



1. Objetivos

- Explicar la organización de la parte práctica de la asignatura.
- Saber utilizar el lenguaje Python para construir aplicaciones.
- Explicar tipos simples y agregados en Python.
- Explicar entrada y salida de datos (y ficheros).



2. Horario

- Grupo **L3**: viernes 11:00 a 13:00, Laboratorio 101
- Grupo **L6**: lunes 12:00 a 14:00, Laboratorio 102

- Profesor: Cristian Tejedor
 - **Contacto**: cristian [at] infor.uva.es

 - Webs de la asignatura:
<https://www.infor.uva.es/~cvaca/asigs/paradigmas.html>
<https://aulas.inf.uva.es/>



3. Entrega de prácticas

- Se tendrá que realizar **2 entregas** de prácticas:
 - Práctica 1: 15% nota
 - ❖ Semana del 27/03/2017 (provisional)
 - Práctica 2: 15% nota
 - ❖ Semana del 15/05/2017 (provisional)
- La parte práctica **no es recuperable** en la convocatoria **extraordinaria**.
 - Si parte teórica en extraordinaria > 3.5 aprobado.



4. El lenguaje Python (I)

- Lenguaje de *script*.
- Permite programación imperativa, orientada a objetos y funcional.
- Programación de interfaces gráficas de usuario.





4. El lenguaje Python (II)

- Página oficial: <https://www.python.org/>
 - Software: intérprete de Python
 - Documentación a utilizar: **versión 2.X** 😊
 - ❖ La versión 3.X está en desarrollo activo.
 - ❖ Pero el software más importante actual no funciona (por norma general) con esta versión 😞
 - ❖ Cambios en la sintaxis: `print("The answer is", 2*2)`

5. Sesiones de laboratorio (I)



- Lectura del libro "Python para todos", Raúl González Duque: <http://mundogeek.net/tutorial-python/>
- Sesión 1 (semana 13-19 febrero):
 - Introducción
 - Mi primer programa en Python
 - Tipos básicos
 - Colecciones
 - Entrada/Salida y ficheros

5. Sesiones de laboratorio (II)



- Sesión 2 (semana 20-26 febrero):
 - Control de flujo
 - Funciones
 - Orientación a objetos (I)
- Sesión 3 (semana 27 fe-5 marzo): trabajar práctica 1.
- Sesión 4 (semana 6 -12 marzo): trabajar práctica 1.
- Sesión 5 (semana 13-19 marzo): trabajar práctica 1.

5. Sesiones de laboratorio (III)



- Sesión 6 (semana 20-26 marzo):
 - Orientación a objetos (II)
 - Interfaces gráficas: pyGTK y Glade
- Sesión 7 (semana 27mar-2 abril): corregir práctica 1.
- Sesión 8 (semana 3-9 abril): Programación funcional.
- Sesión 9 (semana 17-21 abril): trabajar práctica 2.

5. Sesiones de laboratorio (IV)



- Sesión 10 (semana 1-7 mayo): trabajar práctica 2.
- Sesión 11 (semana 8-14 mayo): trabajar práctica 2.
- Sesión 12 (semana 15-21 mayo): corregir práctica 2.

6. Mi primer programa en Python (I)



- Arranque en Linux (aunque si tienes el entorno configurado en otro S.O. sirve también 😊).
- Lenguaje **Python**
 - Distinto a los conocidos: Java, C...
 - Lenguaje **interpretado** (Java es un híbrido compilado-interpretado)
 - **Intérprete** vs compilador
 1. Leído: **por líneas** vs total + trad.
 2. Traducido: **en directo** vs antes
 3. Plataforma: **cualquiera** vs origen
 4. Ejecutable generado: **no** vs sí.
 5. Ejecución: **lenta** vs rápida
 6. **Portabilidad** vs una traducción
 7. Código fuente: **visible** vs oculto
 8. Errores: **ejecución** vs compila.
 9. Parada: **fácil** (control del intérprete) vs difícil (control S.O.).
 - No tiene orden, más rápido y divertido.



6. Mi primer programa en Python (II)

- ¿Cómo trabajar en Python? Varias opciones:

a) Sesión de trabajo (volátil) ☹️

```
weaver7x@weaver7x-X64-VX6:~$ python
Python 2.7.4 (default, Sep 26 2013, 03:20:26)
[GCC 4.7.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
2
>>> exit()
weaver7x@weaver7x-X64-VX6:~$
```

- Si están las dos versiones de Python instaladas: ~\$ `python2`

b) Ficheros de código (como si fuéramos a "compilar") 😊

```
weaver7x@weaver7x-X64-VX6:~$ vi primero.py
weaver7x@weaver7x-X64-VX6:~$ python primero.py
2
weaver7x@weaver7x-X64-VX6:~$ ./primero.py
2
weaver7x@weaver7x-X64-VX6:~$
```

- Si da problemas, poner primera línea del fichero: `#!/usr/bin/python`

6. Mi primer programa en Python (III)



c) Entorno de desarrollo (IDLE, Eclipse)

```
PyDev - pruebapython/paquete1/prueba.py - ADT
```

```
prueba.py  
print 1+1
```

```
Console  
<terminated> /home/weaver7x/workspace/pruebapython/paquete1/prueba.py  
2
```

- Edición y ejecución a la vez 😊😊
- Autocompletado 😊😊
- Revisión de errores 😊😊

6. Mi primer programa en Python (IV)



- Entorno recomendado: Eclipse (última versión)
 - I. Instalar **Python 32bits** <http://www.python.org/>
 - II. Instalar *plugin* "**pydev**" de Eclipse (Neón o superior):
 1. Abrir Eclipse IDE y seleccionar "Help – Install New Software" / "Add".
 2. Poner: "Name": pydev y "Location": **http://pydev.org/updates**
 3. Elegir "PyDev for Eclipse" (*1ª opción, y aceptar todas las licencias / next*).
 4. Reiniciar Eclipse IDE.
 5. Clic en "Window / Preferences / PyDev / Interpreters / Interpreter Python", "new":
 - Name: python2
 - Location /usr/bin/python2 (el mismo en el apartado 1)
 6. Clic en "siguiente" y elegir todas las opciones (excepto los .zip)



7. Tipos básicos (I)

- Tipos de datos (simples, estructurados)
 - **Enteros:** `1`, `3493`
 - **Enteros largos:** `long(1233)`
 - **Octal:** `01`, `06645`
 - **Hexadecimal:** `0x1`, `0xDA5`
 - **Reales:** `67.8` (tener cuidado con los redondeos)
 - **Números complejos:** `1+2i` equivale a: `1+2j`
 - Otra forma: `complex(1,2)` # Sería `1+2j`



7. Tipos básicos (II)

- **Operadores**

- Los de siempre: (+, -, *, +=, -=), pero no existe ++, --
- Exponenciación: 4**2
- División entera: 6/7 #0
- División real:
 - ❑ 6.0 / 7 #0.5
 - ❑ float(6) / 7 #0.5
- División entera con operandos reales:
 - ❑ 12.0 // 7 #1.0
 - ❑ 12.0 / 7 #1.71
- Resto: 12 % 7 # 5



7. Tipos básicos (III)

- Operadores **de bits**
 - `1 & 2 # 01 & 10`, resultado: 0
 - `1 & 3 # 01 & 11`, resultado: 1
- **Cadenas** de caracteres: `" "` o `“ ”` pero no mezclar.
 - `\n` salto de línea o `\t` tabulación, dentro de las comillas
- Operadores **lógicos**: `==` `!=` `<` `<=` `>` `>=` `and` `or` `not`
 - `True` `False`



7. Tipos básicos (IV)

- **Variables**

- Declaración: cualquier lugar.
- Acceso: deben estar declaradas antes.
- No hay que indicar **;** al final como en otros lenguajes.
- No tienen ningún 'tipo' asociado.
- Su 'tipo' es el actual, el que se le asigna.
- El 'tipo' cambia cuando queramos: *tipado dinámico*
 - ❑ `a = 6`
 - ❑ `type(a) # <type 'int'>`
 - ❑ `a = "hola"`
 - ❑ `type(a) # <type 'str'>`



7. Tipos básicos (V)

- `dir()` # Muestra la lista de variables hasta el momento
- `del(a)` # Elimina la variable `a` de mi espacio de nombres. No el contenido de `'a'`, sino `'a'`.
- Mucho cuidado con el **sangrado/espaciado** (uniforme, elegir **espacios** o **tabulaciones**).
- **Subrayado** (sólo para línea de comandos):
 - `iva = 21`
 - `cantidad = 120`
 - `print cantidad * iva / 100 # 25`
 - `cantidad + _ # 145`, devuelve el último valor calculado



7. Tipos básicos (VI)

- Biblioteca *math* (funciones matemáticas)
 - `import math`
 - `math.sqrt(2)` # 1.414, de cualquier otra manera daría error
- Ejecución por línea de comandos (como *script*)
 - \$consola: `which Python` # /usr/bin/Python
 - \$consola: `vi pp.py`
 - ❑ `#!/usr/bin/python`
 - ❑ `print (1+2j)/7`
 - \$consola: `python pp.py`
 - \$consola: `chmod +x pp.py` # se le da permisos de ejecución
 - \$consola: `./pp.py` # se le da permisos de ejecución



7. Tipos básicos (VII)

- Función *print*
 - En el intérprete no hace falta escribirla
 - En ficheros .py sí
 - Es diferente en Python 2.X y 3.X
 - `print "Hola"+"Mundo"` # HolaMundo + salto de línea
 - `print "Hola","Mundo"` # Hola Mundo + salto de línea
 - `print "Hola", "Mundo"` # Hola Mundo + sin salto de línea



8. Colecciones (I)

- Son agrupaciones de datos, ejemplos en Python:

1. **Lista:** `lista = [22, true, "dato", 0]`

- Puede contener listas o cualquier elemento Python.
- Tamaño variable.
- Operaciones: recorrido (hacia delante, hacia atrás), acceso, asignación (***mutable***), añadir, quitar, longitud.
- `type(lista) # <type 'list'>`

2. **Tupla:** `t = (2, 3, 4)`

- Inmutables***
- Puede contener elementos mutables (listas).
- Tamaño fijo.
- `type(t) # <type 'tuple'>`

8. Colecciones (II)



3. Diccionario: `d= {}` `d["pepe"]="amigo"`

- ❑ **Clave** (única): **valor** (cualquier tipo) {Matriz asociativa}
- ❑ Desordenado (en principio).
- ❑ Operaciones: recorrido, ordenación, obtención de claves, asignación (**clave inmutable-valores mutables**), eliminación.
- ❑ `type(lista) # <type 'list'>`

4. Conjunto: `A = set(d)`

- ❑ Se crean a partir de listas, tuplas, o diccionarios.
- ❑ Mismas propiedades de conjuntos matemáticos.
- ❑ Operaciones: `in`, `not in`, `issubset()`, `issuperset()`, `!`, `&`, `-`, `^`, `add()`, `discard()`, `clear()`, copia de conjuntos (por valor y referencia).



9. Entrada/Salida

- Salida de datos:
 - `print "hola" # Variantes con , o con +`
 - `print "Hola %s" % "mundo"`
 - `print "%s %s" % ("Hola", "mundo")`
 - `print r"Hola \n Mundo" # Anula el salto de línea`
 - `print "Hola \\n Mundo" # Anula el salto de línea`
- Entrada de datos: `# import sys sys.argv[0]`
 - `s = raw_input() # Escribimos lo que queremos`
 - `print s, type(s) # Obtenemos el tipo de datos`
 - Podemos restringir la entrada, i.e.: `try-except`



10. Ficheros

- Fichero = objeto de tipo `'file'`.
 - `open(<ruta_nombre_ext>, <modo>, <tamaño_opcional>)`
 - `<modo>`: `'r'` (lectura) `'w'` (escritura) `'a'` (añadir) `'b'` (binario)
`'+'` (lectura/escritura) <<Se pueden combinar>>
 - `f = open("archivo.txt", "r")`
 - `completo = f.read()` # Cadena con todo el contenido de f
 - `parte = f.read(512)` # Cadena con los primeros 512 bytes
 - `linea = f.readline()` # Cadena con la 1ª línea del fichero
 - `f = open("archivo.txt", "w")`
 - `f.write('Esto es una prueba')` #`open(", 'w')` o #`open(", '+')`
 - `f.writelines(["String 1", "String 2"])`
 - `fich.flush()` # Fuerza el guardado inmediato (recomendado)
 - `close(f)` # Importante, siempre cerrar el fichero



Para afianzar...

- Ejercicios de la asignatura (sesión 1) ~30min:
 - <https://www.infor.uva.es/~cvaca/asigs/docpar/sesion1.pdf>
- "Python para todos", Raúl González Duque
 - <http://mundogeek.net/tutorial-python/>
 - **Capítulos:** Introducción, Mi primer programa en Python, Tipos básicos, Colecciones, Entrada/Salida y Ficheros.