Paradigmas de Programación

Sesión 1 Laboratorio

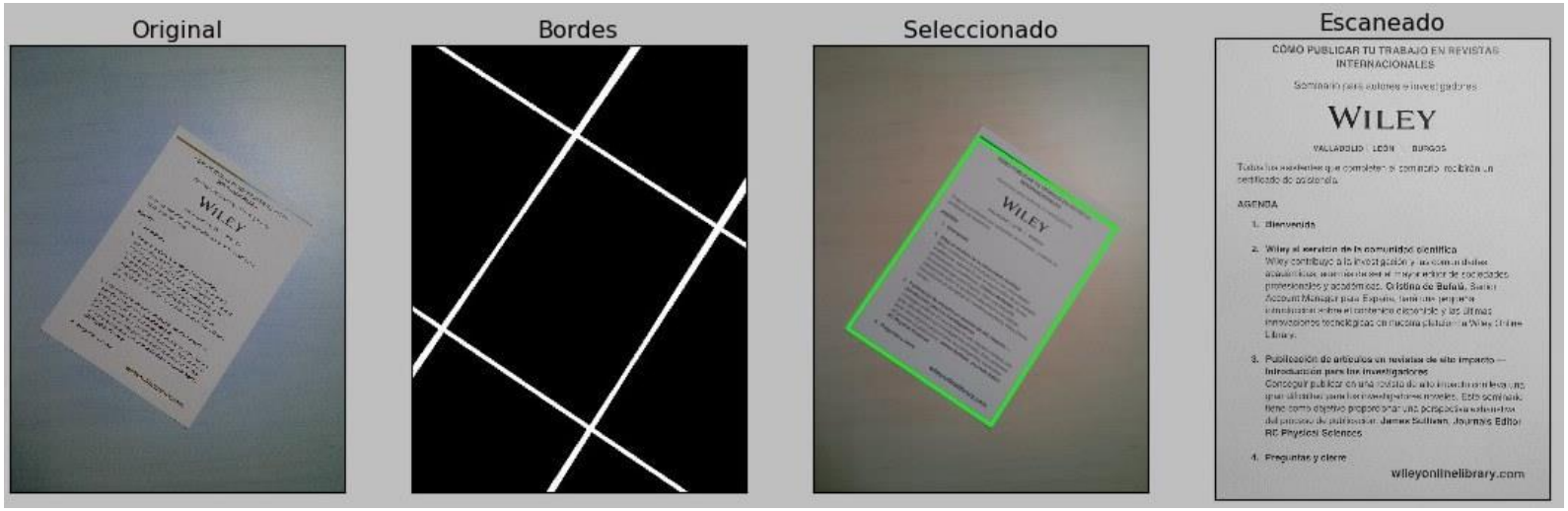
Cristian Tejedor García
Departamento de Informática
Universidad de Valladolid

Curso 2017-18

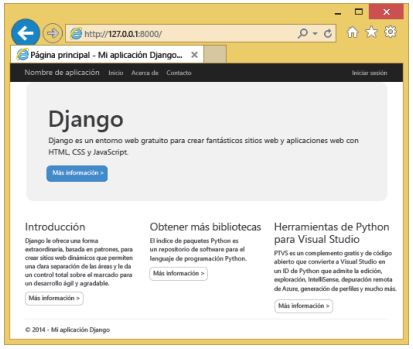
Grado en Ingeniería Informática
Grado en Estadística
INDAT

0. Motivación Python

Potente procesamiento de imágenes (OpenCV)



Web (Django)



Escritorio (PyGTK + Glade)

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.
- Indicar cómo 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 **L8**: martes 10:00 a 12:00, Laboratorio 101
- Grupo **L4**: viernes 11:00 a 13:00, 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 26/03/2017 (provisional)
  - Práctica 2: 15% nota
    - Semana del 14/05/2017 (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 2.7.X** ☺
    - La **versión 3**.6.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 12-18 febrero):
  - *Introducción*
  - *Mi primer programa en Python*
  - *Tipos básicos*
  - *Colecciones*
  - *Entrada/Salida y ficheros*

5. Sesiones de laboratorio (II)

- Sesión 2 (semana 19-25 febrero):
  - *Control de flujo*
  - *Funciones*
  - *Orientación a objetos (I)*
- Sesión 3 (semana 26 febrero - 4 marzo):
  - *Cadenas de caracteres (Strings)*
- Sesión 4 (semana 5-11 marzo): trabajar práctica 1.
- Sesión 5 (semana 12-18 marzo): trabajar práctica 1.

5. Sesiones de laboratorio (III)

- Sesión 6 (semana 19-25 marzo): **corregir práctica 1**.
- Sesión 7 (semana 9-15 abril):
  - *Orientación a objetos (II)*
  - *Interfaces gráficas: pyGTK y Glade*
- Sesión 8 (semana 23-29 abril): Programación funcional.
- Sesión 9 (semana 30-6 mayo): trabajar práctica 2.

5. Sesiones de laboratorio (IV)

- Sesión 10 (semana 7-13 mayo): trabajar práctica 2.
- Sesión 11 (semana 14-20 mayo): trabajar práctica 2.
- Sesión 12 (semana 21-27 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

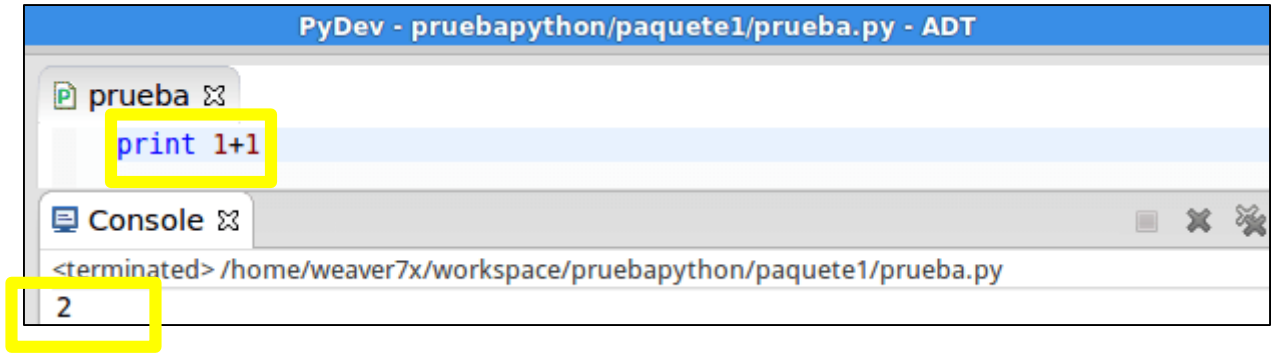
  b) Ficheros de código (como si fuéramos a "compilar") ☺

```
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 (IDE: Eclipse, PyCharm)



- Edición y ejecución a la vez ☺☺
- Autocompletado ☺☺
- Revisión de errores ☺☺
- Visionado de documentación 'in-situ' ☺☺

6. Mi primer programa en Python (IV)

- Entorno recomendado: **Eclipse** portable (última versión)
  I. Instalar **Python v. 2.7. 32bits** http://www.python.org/
  II. Instalar *plugin* "**pydev**" de Eclipse (Neón o superior):
    1. Abrir Eclipse IDE: "Help – Install New Software" / Pulsar en "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 / Python Interpreter" y pulsar en "new":
       - Name: **python2**
       - Location **/usr/bin/python2 Linux** (en **Windows <*ruta*>/python.exe**)
    6. Clic en "siguiente" y elegir todas las opciones (excepto los .zip en Windows si aparecieran, ya que en Linux no aparecen).

6. Mi primer programa en Python (V)

- Si tras el paso 4 del segundo punto de la diapositiva anterior no se ve el plugin 'pydev', es porque **no** se tiene una versión de **Eclipse compatible**.

- Existen dos soluciones:

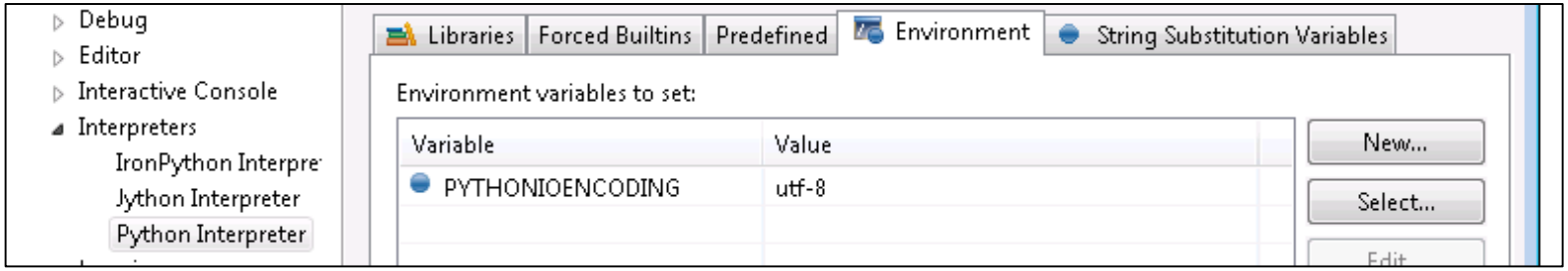
  A. **Instalar** la última versión de Eclipse (o la versión portable) y realizar de nuevo los pasos de la diapositiva anterior.

  B. **Mantener** la versión antigua de Eclipse:

  1. Descargar el fichero pydev 2.8.2 de:
     - https://sourceforge.net/projects/pydev/files/pydev/
  2. Cerrar Eclipse.
  3. Extraer las dos carpetas del fichero descargado en el directorio de Eclipse (reemplazar ambas).

6. Mi primer programa en Python (VI)

- Es posible que trabajando con Windows, Eclipse y Pydev, a la hora de trabajar con los caracteres unicode: `print u'\u2500', u'\u2502'  # — |` Eclipse encuentre un problema de codificación al ejecutar o incluso grabar.
- Los pasos que pueden solucionarlo son los siguientes:
  1. En Eclipse, seleccionar:
     - Window -> Preferences -> PyDev -> **Interpreters** (zona izquierda) -> Python Interpreter -> **Environment** (pestaña) -> New (botón)
  2. Escribir:
     - **Name: PYTHONIOENCODING**
     - **Value: utf-8**
- (el campo Name: debe estar escrito en mayúsculas), de forma que, una vez aceptado, el aspecto será el de la figura siguiente:



- Además, debe escribirse en la primera línea de cada fichero .py: `# -*- coding: utf-8 -*-`

6. Mi primer programa en Python (VII)

- Otro IDE recomendado: **PyCharm**

1. Instalar **Python v. 2.7.** 32bits http://www.python.org/
2. Instalar la versión **Community** de Pycharm:

   https://www.jetbrains.com/pycharm/download/

7. Tipos básicos (I)

- Tipos de datos (simples, estructurados)
  - **Enteros**: 1, 3493
  - **Enteros largos**: ***long***(1233)
  - **Octal**: **0**1, **0**6645
  - **Hexadecimal**: **0x**1, **0x**DA5
  - **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 mezclarlas.
  - \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'*, si no *'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 # from math import sqrt
  - math.sqrt(2) # 1.414
- 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 hace falta escribirla
  - Es diferente en Python 2.X y 3.X (sin/con paréntesis)
  - 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, da igual el tipo.
     - Tamaño variable.
     - Operaciones: recorrido (hacia delante, hacia atrás), acceso, asignación (***mutable***), adición, eliminación, inversión, longitud…
     - type(lista) # <type 'list'>
  2. **Tupla**: t = (2, 3, 4)
     - ***Inmutables***
     - Puede contener elementos mutables (listas).
     - Tamaño fijo: mayor rendimiento en memoria.
     - type(t) # <type 'tuple'>

8. Colecciones (II)

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

- **Clave** (única): **valor** (cualquier tipo) {Matriz asociativa, mapa}
- 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 queramos
  - 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: i.e.: 'ab' >>
  - f = open("archivo.txt", "r") # with open("archivo.txt", "r") as f
  - 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') # abrir con *open('','w')* o *open('','+')*
  - f.writelines(["String 1", "String 2"])
  - f.flush() # Fuerza el guardado inmediato (recomendado)
  - f.close() # 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.