Paradigmas de Programación

Iniciación a Python3, variables, listas, tuplas, diccionarios, conjuntos, ficheros

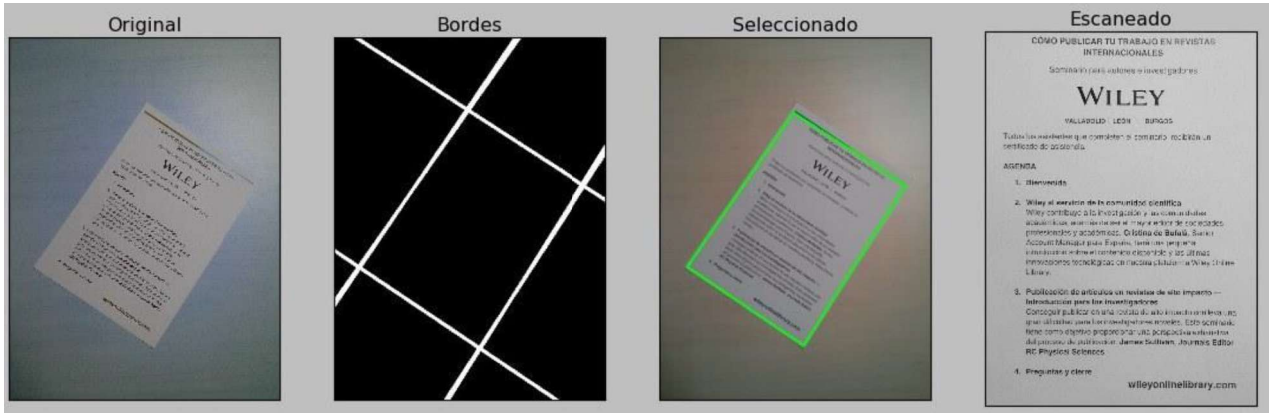
Cristian Tejedor García
Departamento de Informática
Universidad de Valladolid

Curso 2025-26

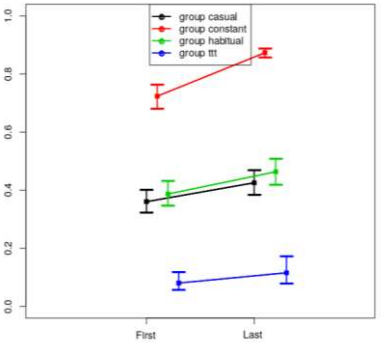
Grado en Ingeniería Informática
Grado en Estadística
INDAT

0. Motivación Python

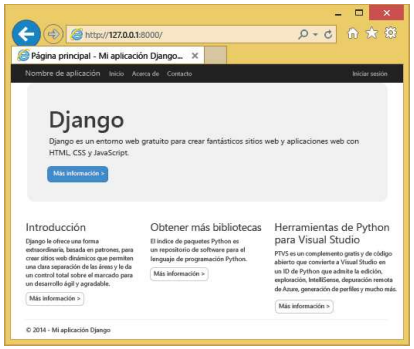
Potente procesamiento de imágenes (OpenCV)



Análisis estadístico (Pandas, NumPy, SciPy…)



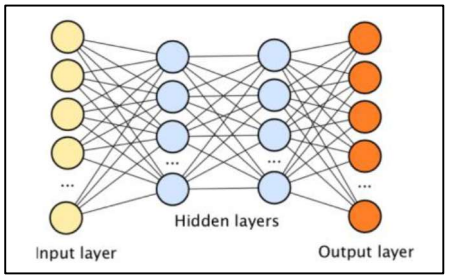
Web (Django)



Escritorio (wxPython)

Scripting

Machine learning (DNN)



Contenido

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).

4. El lenguaje Python (I)

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

4. El lenguaje Python (II)

- Página oficial: https://www.python.org/
  - Software: intérprete de Python 3
  - Documentación a utilizar: **versión 3.6.8** ☺
    - La **versión 2.X** ya no está soportada.
    - https://www.python.org/doc/sunset-python-2/

5. Sesiones de laboratorio (I)

- Lectura del libro "Python para todos", Raúl González Duque: http://www.utic.edu.py/citil/images/Manuales/Python_para_todos.pdf

- Sesión 1:
  - *Introducción*
  - *Mi primer programa en Python*
  - *Tipos básicos*
  - *Colecciones*
  - *Entrada/Salida y ficheros*

5. Sesiones de laboratorio (II)

- Sesión 2:
  - *Control de flujo*
  - *Funciones*
  - *Orientación a objetos (I)*
- Sesión 3:
  - *Cadenas de caracteres (Strings)*
- Resto de sesiones hasta entrega: trabajar práctica 1.

5. Sesiones de laboratorio (III)

- Sesión 7:
  - *Orientación a objetos (II)*
  - *Interfaces gráficas: wxWidgets, wxPython, wxGlade*
- Sesión 8:
  - *Programación funcional*
- Resto de sesiones hasta entrega: trabajar 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) ☹

```
Python 3.6 (32-bit)
weaver7x@weaver7x-X64-VX6:~$ python3
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 23 2018, 23:31:17) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
2
>>> exit()
```

  - *Si están las dos versiones de Python instaladas*: ~$ python3

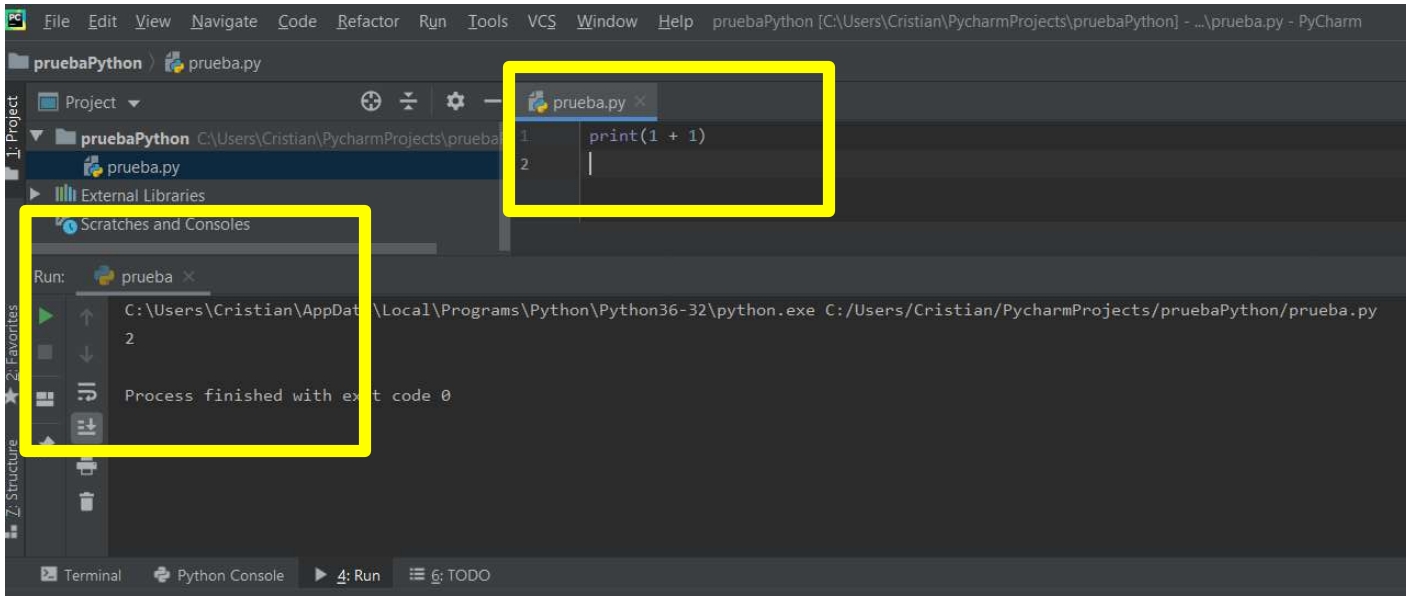
  b) Ficheros de código (como si fuéramos a "compilar") ☺

```
weaver7x@weaver7x-X64-VX6:~$ vi primero.py
weaver7x@weaver7x-X64-VX6:~$ python3 primero.py
2
weaver7x@weaver7x-X64-VX6:~$ ./primero.py
2
weaver7x@weaver7x-X64-VX6:~$
```

  - *Si da problemas -> primera línea del fichero*: #!/usr/bin/python3

6. Mi primer programa en Python (III)

c) Entorno de desarrollo (IDE: 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: PyCharm

  PyCharm 2019.3 (**Community** version)

Linux: *desde consola escribir:*

```
sudo apt-get update
sudo apt-get install python3.6
snap install --classic pycharm-community
```

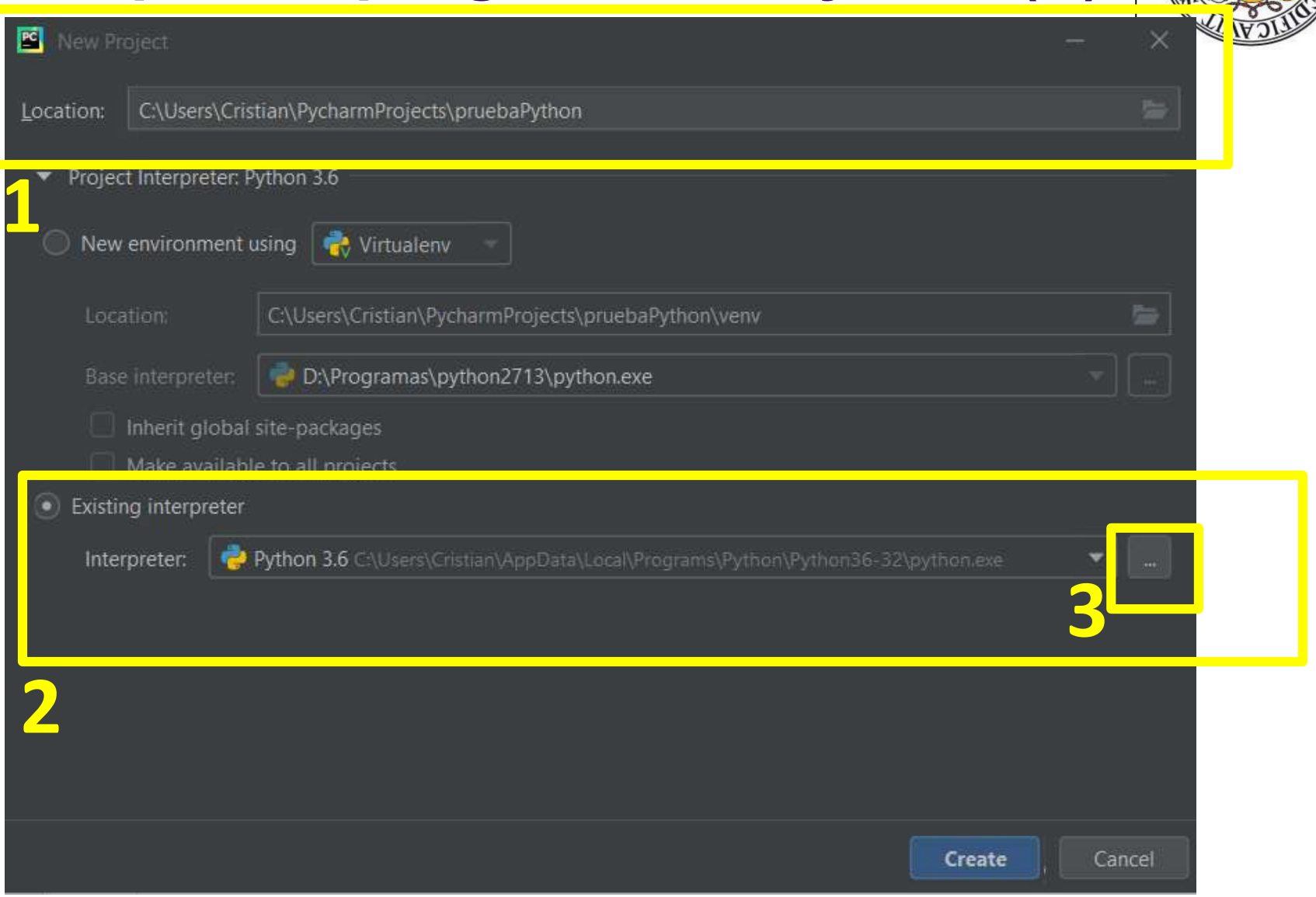
O desde el Centro de Software:

Windows:

I. Instalar **Python v. 3.6.8 - 32bits** + activar todas las opciones de añadir al *path* en el diálogo de instalación: https://www.python.org/downloads/release/python-368/

II. Instalar **PyCharm 2019.3** (Community version): https://www.jetbrains.com/pycharm/download

6. Mi primer programa en Python (V)



6. Mi primer programa en Python (VI)

- Otro IDE recomendado: Visual Studio Code

1. Instalar **Python v. 3.6.8** 32bits + activar todas las opciones de añadir al *path* en el diálogo de instalación:

   https://www.python.org/downloads/release/python-368/

2. Instalar **V**isual **S**tudio **C**ode (Community version):

   https://code.visualstudio.com/download

3. Instalar extensión de Python dentro de VSC.

7. Tipos básicos (I)

- Tipos de datos simples (*variable* = *valor*)
  - **Enteros (int)**: `a = 1, b = 3493` # En Python3 no existe ***long***
  - **String** (cadena de carácter): `cadena = "Hola"`
  - **Transformar de cadena a entero:** `int("56")`
  - **Octal**: `o1 = 0o1, o2 = 0o6645` # También **oct**(numero)
  - **Hexadecimal**: `h1 = 0x1, h2 = 0xDA5` # También **hex**(numero)
  - **Reales**: `r1 = 67.8` # Cuidado con los redondeos
  - Números **complejos**: `1+2i` equivale a: `c1 = 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 / 7 #0.8571428571428571
    - float(6) / 7 #0.8571428571428571
  - División entera con operandos reales:
    - 12.0 // 7 #1.0
    - 12.0 / 7 #1.7142857142857142
  - 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
    - print(type(a)) # <class 'int'>
    - a = "hola"
    - print(type(a)) # <class '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** (siempre debe ser uniforme: tabulaciones o espacios).
- **Subrayado** (sólo para línea de comandos):
  - iva = 21
  - cantidad = 120
  - print(cantidad * iva /100) # 25.2
  - print(cantidad + _) # 145.2,
    - Devuelve el último valor calculado.

7. Tipos básicos (VI)

- Biblioteca *math* (funciones matemáticas)
  - import math # Otra opción: from math import sqrt
  - print(math.sqrt(2)) # 1.4142135623730951
- 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",end="") # Hola Mundo + sin salto de línea

8. Colecciones (I)

- Son agrupaciones de datos, ejemplos en Python:
  1. **Lista**: lista = [22, True, "dato", 0, [1,3,4]]
     - Puede contener **valores** de **cualquier tipo** de **datos**.
     - Tamaño variable.
     - Operaciones: recorrido (hacia delante, hacia atrás), acceso, asignación (***mutable***), adición, eliminación, inversión, longitud…
     - print(type(lista)) # <class 'list'>
  2. **Tupla**: t = (2, 3, 4)
     - ***Inmutables***
     - Puede contener elementos mutables (listas).
     - Tamaño fijo: mayor rendimiento en memoria.
     - print(type(t)) # <class '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.
- print(type(d)) # <class 'dict'>

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).
- print(type(A)) # <class 'set'>

9. Entrada/Salida

- **Salida** de datos:
  - print("hola") # Variantes de los parámetros:
  - print("Hola" + "mundo")
  - print("Hola", "mundo")
  - 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 = 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:  # sin close()
frases = f.readlines() # Lista cuyos elementos son cada línea del fichero
```

  - Otras opciones menos recomendables:
    - completo = f.read() # Cadena con todo el contenido de *f*
    - linea = f.readline() # Cadena con la 1ª línea del fichero. Ejemplo

```
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, tanto en **lectura** como en **escritura**

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://www.utic.edu.py/citil/images/Manuales/Python_para_todos.pdf
  - **Capítulos**: Introducción, Mi primer programa en Python, Tipos básicos, Colecciones, Entrada/Salida y Ficheros.