

1. Herramientas

Introducción a Python para Deep Learning

JUNTA DE EXTREMADURA
Consejería de Economía, Empleo y Transformación Digital

ETD
EXTREMADURA
Estrategia de Transformación
Digital de Extremadura




uex



Contenidos

1. Consola
2. Google Colab
3. JupyterLab
4. Visual Studio Code
5. Entornos virtuales
6. ¡Ayuda!
7. Docker

1. Consola

- Abrid un terminal
 - En Windows puede ser  + r
 - **cmd**, **powershell** o **terminal**
 - En Linux y macOS tenéis el terminal
- Escribid python
- Escribid el código a ejecutar: `print("Hola, mundo")`
- También podemos usar archivos
- Guardad en un archivo de texto el código anterior
- Escribid python `hola.py`

1. Consola

¿Ventajas?

¿Inconvenientes?

1. Consola

- Ventajas
 - rápido para probar
 - simple
 - uso mínimo de recursos
 - siempre disponible: si tienes Python, está
 - accesible desde conexiones remotas
 - fácil de usar para tareas básicas



1. Consola

- Inconvenientes
 - árido, rústico, rudimentario
 - sin persistencia
 - complicado de modularizar
 - falta de herramientas avanzadas (que aún no conocéis)
 - salida limitada (sólo texto)



1. Consola

¡Seguro que hay mejores alternativas!



2. Google Colab

- Para programar con Python sólo necesitamos acceso a Internet y un cliente web
- Visítad: <https://colab.research.google.com>
- Usad el mismo código que desde la consola
- Ejecución desde la web
- Estamos utilizando notebooks
- Combinan código y texto
- Resultados persistentes
- Se pueden compartir programas con una URL
- Uso de GPU y TPU



2. Google Colab

```
from PIL import Image  
  
image = Image.open("usc.png")  
image
```

- Probadlo desde Google Colab
- Probadlo desde la consola
- ¿Ventajas?
- ¿Inconvenientes?

2. Google Colab

¿Ventajas?

¿Inconvenientes?

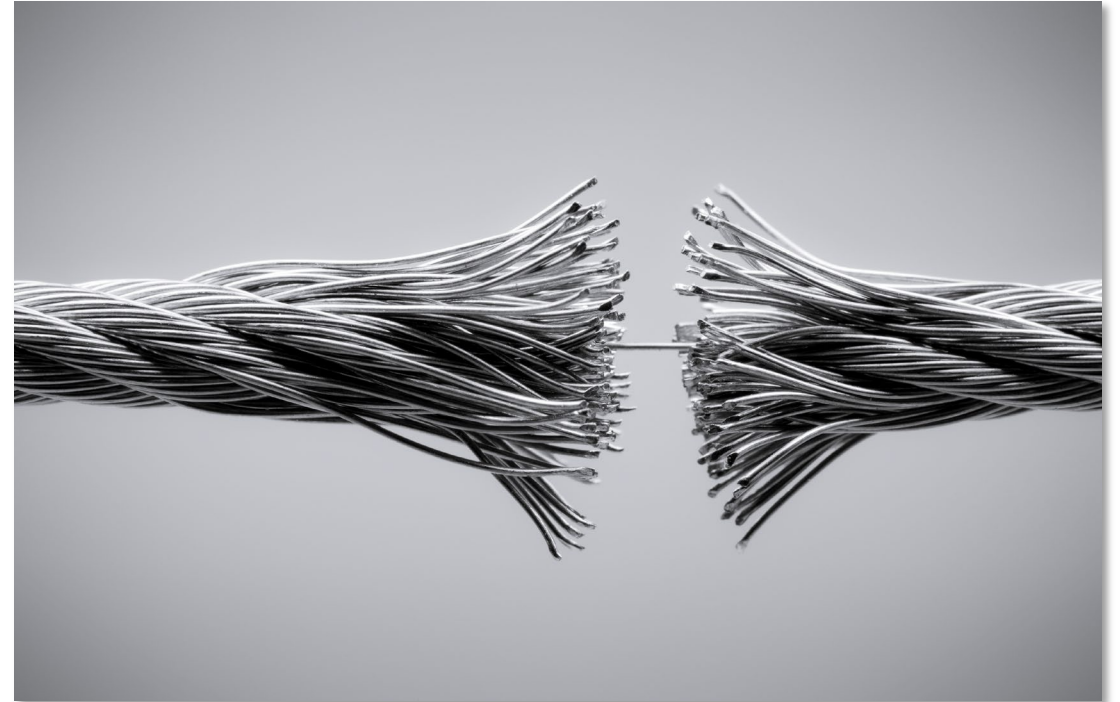
2. Google Colab

- Ventajas
 - Facilidad de acceso
 - Configuración nula
 - Bueno, casi nula
 - Entorno compartido y colaborativo



2. Google Colab

- Desventajas
 - Necesitas conexión a Internet
 - Recursos limitados
 - Menor control



2. Google Colab

Esto está mejor...
¡Pero espero que haya mejores
alternativas!



3. Visual Studio Code

- VSC de ahora en adelante
- Herramienta gratuita, multiplataforma y de código abierto
- Desarrollada por Microsoft 🤖
- Editor de texto extensible
- Con extensiones se puede transformar en un IDE: *Integrated Development Environment* (entorno integrado de desarrollo)
- Nos proporciona todas las herramientas que necesitamos en un único lugar
- Es lo que vamos a usar en clase
- Debería estar instalado en el laboratorio
- Instaladlo en vuestros equipos personales para seguir el trabajo en casa
- <https://code.visualstudio.com/>



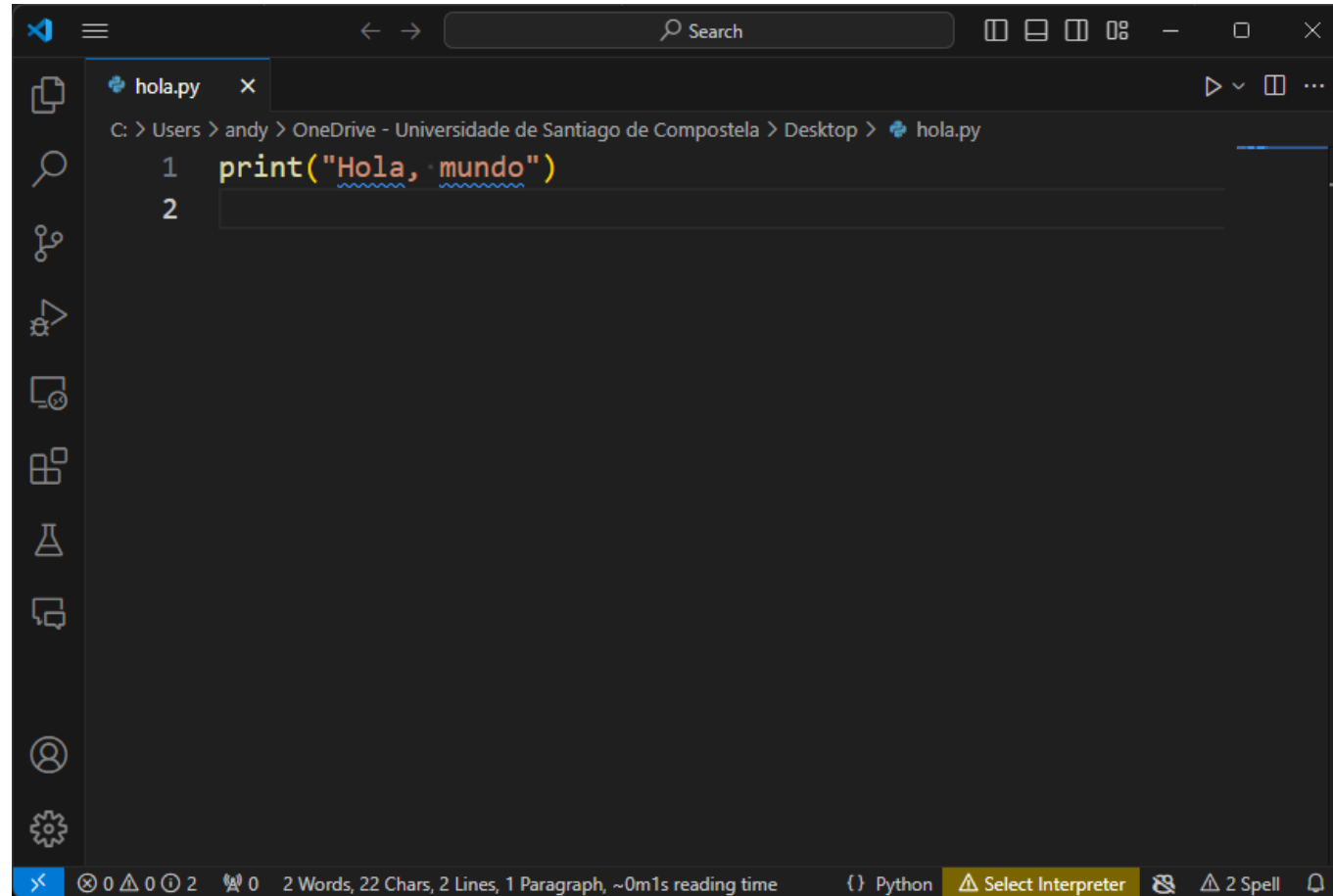
Logo de Visual Studio Code

3. Visual Studio Code

- Nos ayuda en todo lo que puede
- Resalte de sintaxis (colores por tipo de palabra)
- Autocompletar: tanto funciones como variables
- Rodear cadenas es fácil (dobles comillas, paréntesis...)
- Nos avisa de los errores
- Nos muestra el tipo de las variables
- Nos muestra su contenido
- ¡Esto es una maravilla!



3. Visual Studio Code



3. Visual Studio Code

- Cread un nuevo archivo **hola.py** (o abrid el que ya teníamos)
- VSC ofrecerá instalar las extensiones necesarias si no lo están ya
- A partir de ese momento, podremos ejecutar el script
- ¿Y qué se usa para la ejecución...?
- ¡La consola!



3. Visual Studio Code

- ¡Pero podemos hacer mucho más!
- Depuración, por ejemplo
- Probemos a guardar el mensaje en una variable (ya explicaremos en más detalle qué son):

```
mensaje = "Hola, mundo"  
print(mensaje)
```

3. Visual Studio Code

- En lugar de ejecutar, depurar
- Añadir un punto de interrupción antes de `print()`
- Panel de inspección a la izquierda
- Vemos las variables
- Podemos ver sus tipos



3. Visual Studio Code

¿Ventajas?

¿Inconvenientes?

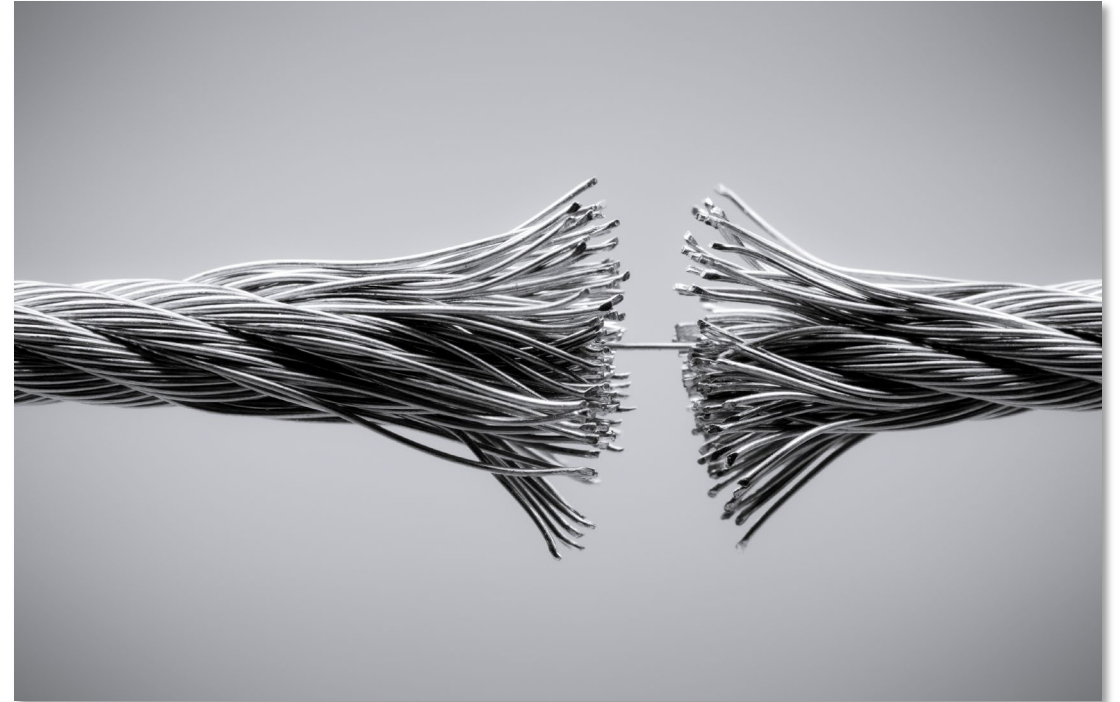
3. Visual Studio Code

- Ventajas
 - Todo lo que hemos comentado anteriormente como sus características



3. Visual Studio Code

- Desventajas
 - No disponible en situaciones en las que no puedas acceder a una interfaz gráfica (conexiones remotas)
 - Existen herramientas más potentes (PyCharm)
 - Una vez te acostumbras a lo bueno...



4. JupyterLab

- <https://jupyter.org/>
- Como Google Colab, ¡pero en nuestro ordenador!
- Bueno, realmente es al revés: Colab es como JupyterLab
- El concepto es tan potente que hasta Apple lo ha copiado:
- <https://developer.apple.com/swift-playground/>

```
pip install jupyterlab  
jupyter lab
```

Reflexión

- ¿Qué versión de Python tenéis?
 - Podría ser la 3.12 o superior
- ¿Qué versión de Python tengo yo?
 - Podría ser la 2.7
- ¿Qué puede pasar?
 - Que un programa que a mí me funcione, a vosotros y vosotras no
 - Que no seamos capaces de trabajar en equipo



Reflexión

- Es **FUNDAMENTAL** que usemos las mismas versiones
- Cuando trabajamos en equipo
- Cuando trabajamos solos, pero el proyecto se prolonga en el tiempo
- Cuando intentamos usar programas desarrollados por otras personas



Reflexión

- Problema: dependencias
- Un programa usa una serie de bibliotecas
- Una biblioteca tiene una vida y una serie de versiones
- Un programa es una combinación de código y bibliotecas
- Resultado:
 - cada programa pertenece a un momento en el tiempo
 - puede que el programa funcione perfectamente con la versión X de una biblioteca, pero con la versión Y no

Reflexión

La solución: los entornos virtuales

5. Entornos virtuales

- Permiten trabajar con diferentes versiones de Python
- Guardan en una carpeta diferente cada entorno
- Contienen todo lo necesario (paquetes estándar y de terceros)
- Se puede activar, desactivar, borrar
- ¿Ventajas y desventajas?



5. Entornos virtuales

- Versionado semántico: significado
- <https://semver.org/>
- Instalar miniconda: <https://docs.anaconda.com/miniconda/> (debería estar ya instalado)
- Existen muchas otras alternativas, no es necesario limitarse a Miniconda
 - venv: <https://docs.python.org/3/library/venv.html> (nativo de Python)
 - Conda: <https://anaconda.org/anaconda/conda>
 - pyenv: <https://github.com/pyenv/pyenv>
- Pero miniconda es la que más nos gusta

5. Entornos virtuales

- Creación de entornos
 - `conda create --name prueba python=3.12 --yes`
- Activación de entornos
 - `conda activate prueba`
- Desactivación de entornos
 - `conda deactivate`
- Eliminación de entornos
 - `conda remove --name prueba --all`

5. Entornos virtuales

- Uso del entorno virtual desde VSC
- **Search** (arriba, en el centro) >**python:s**
- Seleccionar el entorno virtual que necesitemos en cada momento
- Exportar los entornos a un archivo para compartirlos
- Guardarlos junto con el código fuente del proyecto:

```
conda env export > project_environment.yml
```

```
conda env create --file project_environment.yml
```

6. Ayuda

- Documentación de Python
 - Desde la web:
<https://docs.python.org/>
 - Desde el entorno de desarrollo:
poner el puntero encima
- Google: búsquedas
- Stack Overflow: resolución de problemas
- ChatGPT: 😊



6. Ayuda

- ¿Cómo usar ChatGPT? Con cabeza
- Pedirle la solución a un problema:
 - corto plazo
- Pedirle que analice nuestra solución:
 - largo plazo
- *“Dale un pez a un hombre y comerá hoy. Enséñale a pescar y comerá el resto de su vida”*



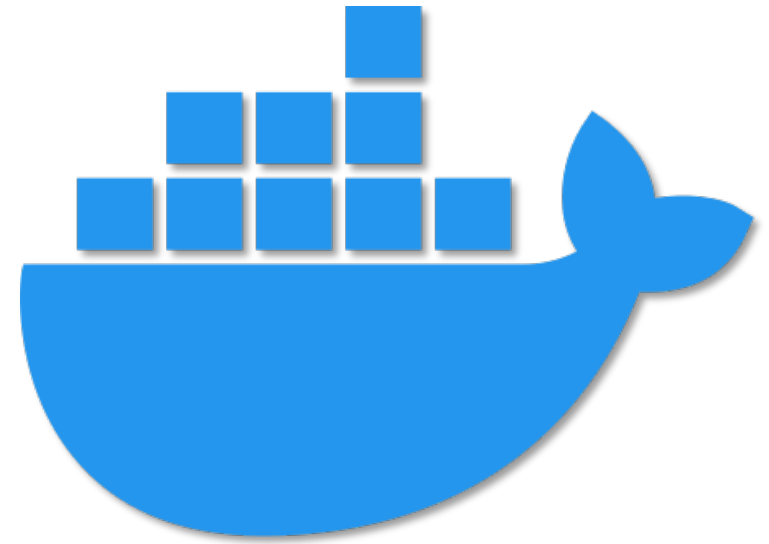
6. Ayuda

- ¿Cómo usar ChatGPT? Con cabeza
- Pedirle la solución a un problema:
 - corto plazo
- Pedirle que analice nuestra solución:
 - largo plazo
- *“Dale un pez a un hombre y comerá hoy. Enséñale a pescar y comerá el resto de su vida”*



7. Docker

- Le dedicaremos más tiempo
- Plataforma que permite ejecutar aplicaciones en entornos aislados llamados **contenedores**
- Permite instalar y ejecutar aplicaciones complejas, como gestores de bases de datos o servidores web, sin preocuparte por las configuraciones en tu sistema operativo
- Fácil de usar
- Multiplataforma



7. Docker

- Docker Desktop
- Debería estar instalado en vuestros equipos
- No lo usaremos en este curso más allá de para explicar su funcionamiento y hacer pruebas, pero sí se usará en otros cursos del programa
- Descargar: <https://www.docker.com/products/docker-desktop/>
- Aplicación de escritorio con interfaz gráfica
- Una vez instalado, lanzar como una aplicación más

7. Docker

Gestión de imágenes:

1. Haz clic **Images**, en el lateral
2. Utiliza el cuadro de texto de búsqueda para encontrar una imagen (por ejemplo, busca **mysql**)
3. Haz clic en **Pull** para descargar la imagen
4. Haz clic en **Run** para ejecutar la imagen

También puedes detener o desinstalar las imágenes

Conclusiones

- Existen múltiples formas de usar Python: consola, notebooks en Colab o en JupyterLab, IDEs...
- Cada una con sus ventajas e inconvenientes
- Usar un IDE es lo ideal
- Combinarlo con entornos virtuales
- Utilizar sabiamente la ayuda a vuestra disposición



Recursos y referencias

- [Google Colab](#)
- [Visual Studio Code](#)
- [JupyterLab](#)
- [Versionando semántico](#)
- [Miniconda](#)
- [Ayuda de Python](#)
- [Stack Overflow](#)
- [ChatGPT](#)
- [Docker](#)
- [Docker Desktop](#)

1. Herramientas

Introducción a Python para Deep Learning

JUNTA DE EXTREMADURA
Consejería de Economía, Empleo y Transformación Digital

ETD
EXTREMADURA
Estrategia de Transformación
Digital de Extremadura



uex

