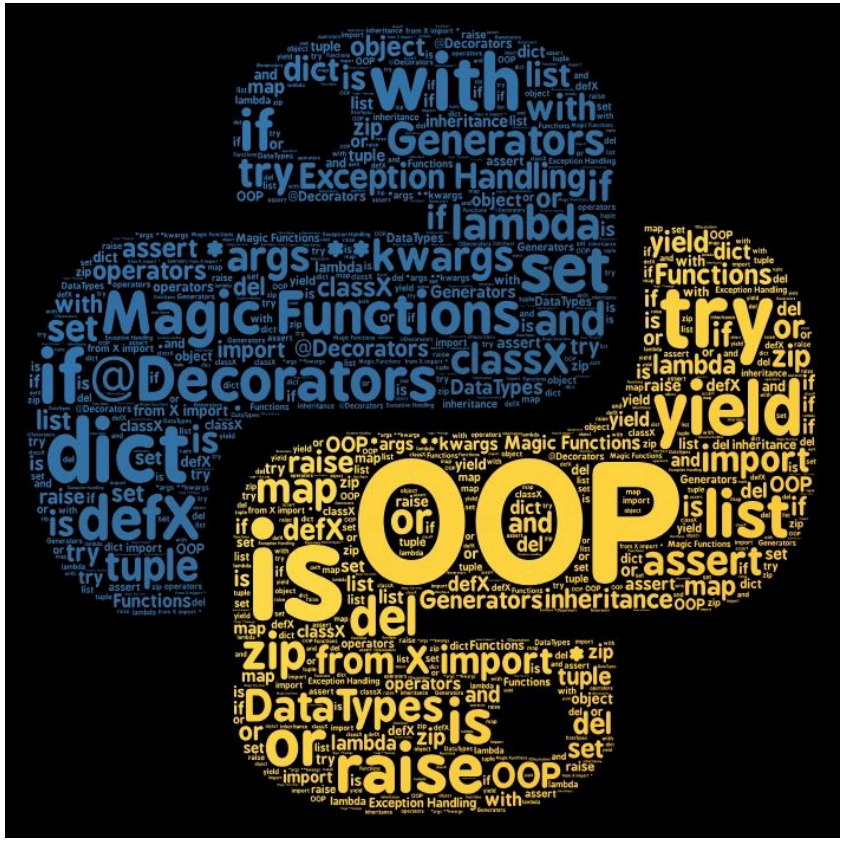
VNIVERSITAT DE VALÈNCIA

ISP · Image & Signal Processing



Introducción a Python

Máster en Sociedad Digital

Oscar José Pellicer Valero

`Oscar.Pellicer@uv.es`

Departament d'Enginyeria Electrònica
Escola Tècnica Superior d'Enginyeria

Programa del curso: Introducción a Python

Horario: Viernes de 15:30h a 18:30h
Requisitos: Trae tu propio portátil!
Sesiones:

- **Sesión 1 (Viernes, 7 de noviembre): Gramática básica y estructuras de datos.** Se sientan las bases de la programación en Python, desde la configuración del entorno con Jupyter hasta el trabajo con variables, texto, y las estructuras de datos fundamentales como listas y diccionarios. Se introducen las funciones como bloques de código reutilizables.
- **Sesión 2 (Viernes, 14 de noviembre): Flujo de control y manejo de ficheros.** Se dominan los elementos para controlar el flujo de un programa, como los bucles y condicionales. Se exploran técnicas más avanzadas y se finaliza con la lectura y escritura de ficheros del sistema.
- **Sesión 3 (Viernes, 21 de noviembre): Análisis y manipulación de datos con Pandas.** Introducción a Pandas, la librería clave para la ciencia de datos. Se aprenderá el ciclo completo del trabajo con datos: carga, inspección, limpieza, filtrado, transformación, y la realización de agregaciones y agrupaciones (`groupby`) para responder preguntas.
- **Sesión 4 (Viernes, 5 de diciembre): Visualización de datos con Matplotlib y Seaborn.** Se aprende a crear visualizaciones de datos profesionales, usando Matplotlib como base y Seaborn para producir gráficos estadísticos atractivos y listos para una presentación.
- **Sesión 5 (Viernes, 12 de diciembre): Machine learning y proyecto final.** Introducción a los conceptos de "machine learning" con Scikit-learn, construyendo un modelo predictivo simple. Se consolidan todas las habilidades aprendidas con un proyecto final de análisis de datos.

Introducción a `conda`, `mamba`, `venv`

Prompt: A toolbox for data scientists, data all over the place

Desinstalando Anaconda

Si ya tienes instalada una versión de Anaconda, para evitar conflictos, es muy recomendable desinstalarla por completo antes de empezar:

Windows:

1. **Usar el desinstalador:** Busca "Agregar o quitar programas", encuentra Anaconda en la lista y haz clic en "Desinstalar". En la carpeta de instalación de Anaconda (ej. `C:\Users\username\Anaconda3`) suele haber un `Uninstall-Anaconda3.exe`.
2. **Limpieza manual:** Después de desinstalar, borra las carpetas residuales como `.conda` y `.continuum` en tu directorio de usuario (`C:\Users\username`).

macOS / Linux:

1. **Eliminar la carpeta de instalación:** Abre una terminal y elimina el directorio de Anaconda: `rm -rf ~/anaconda3` (la ruta puede variar, p.ej., `~/opt/anaconda3`).
2. **Limpiar archivos de configuración:** Elimina las carpetas y archivos de configuración ocultos en tu directorio de inicio: `rm -rf ~/.condarc ~/.conda ~/.continuum`.
3. **Editar el perfil del shell:** Abre tu archivo de configuración del shell (p.ej., `~/.bashrc`, `~/.zshrc`) y elimina la línea que añade Anaconda a tu `PATH`.

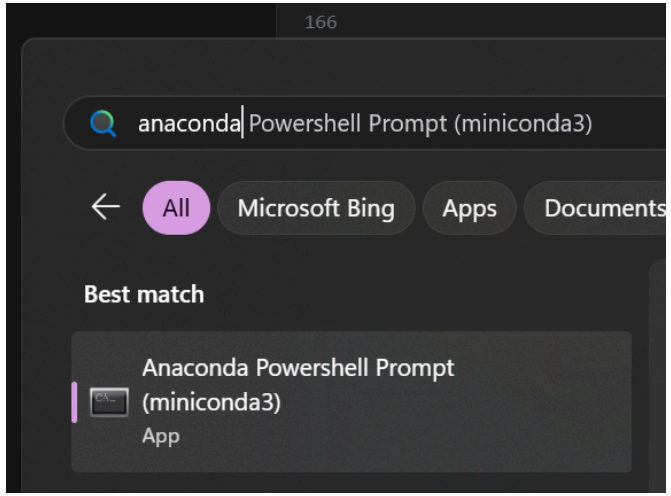
Instalando Miniconda

Miniconda es una versión mínima de Anaconda que incluye solo Python, `conda` y un pequeño número de paquetes. Esto nos da un mayor control sobre nuestros entornos.

Para instalar Miniconda, accede a la página de descarga, haz click en `Get Started`, escoge la opción adecuada para tu sistema operativo, descarga el instalador y instálalo (deja todas las opciones por defecto).

Una vez instalado, para empezar a trabajar con Anaconda:

- En Windows, abre el Anaconda Prompt desde el menú de inicio (ver imagen de la derecha →)
- En macOS, abre la terminal haciendo `Command + Space` y buscando `Terminal`
- En Linux, abre la terminal haciendo `Ctrl + Alt + T`



Instalando Visual Studio Code

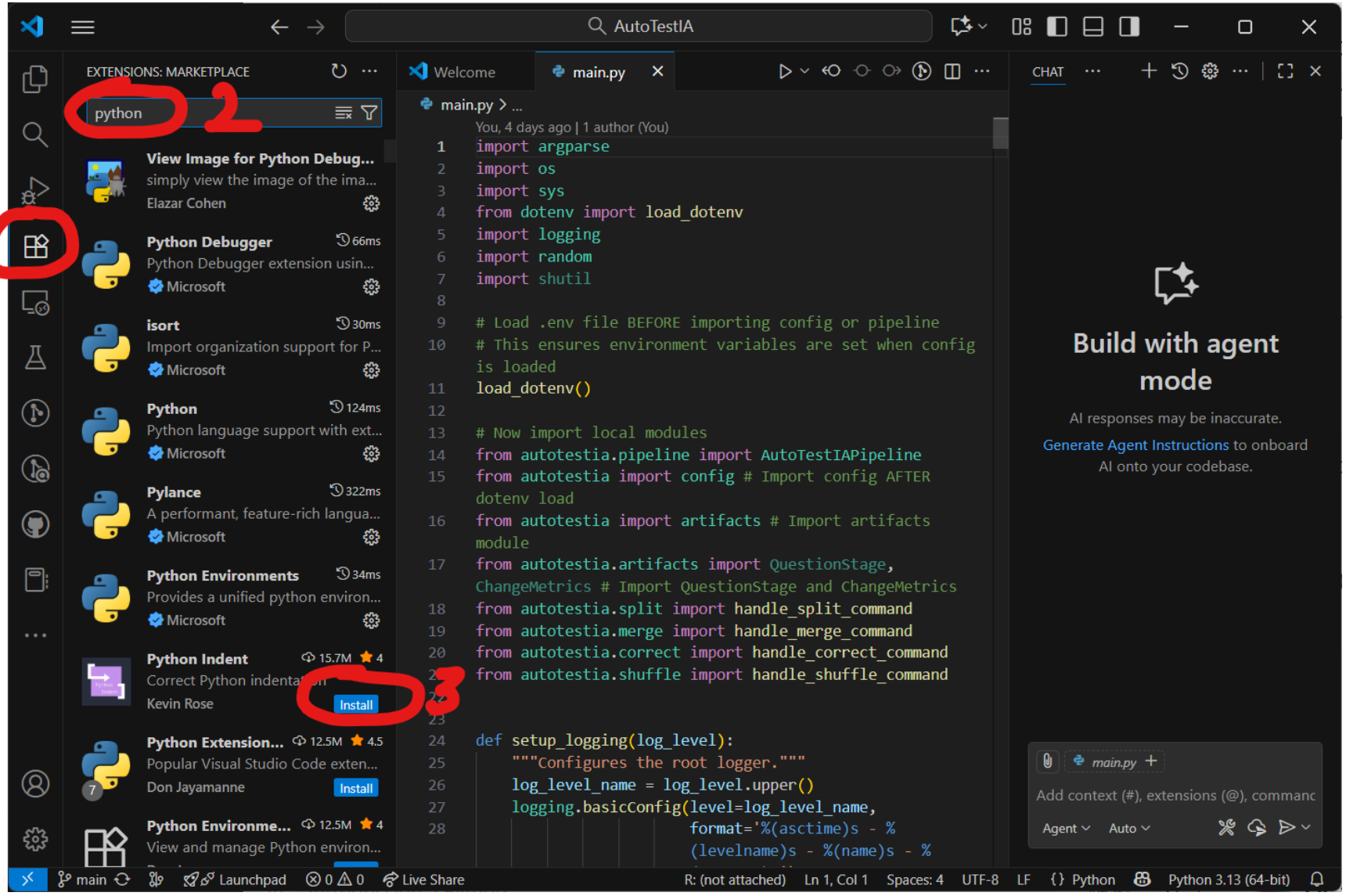
Visual Studio Code (VS Code) es un editor de código ligero y potente que usaremos como nuestra herramienta principal para trabajar con Python y Jupyter Notebooks.

1. **Descarga e instala VS Code:**
   Ve a la web oficial de VS Code y descarga el instalador para tu sistema operativo.
   Sigue las instrucciones de instalación.

2. **Instala las extensiones esenciales:**
   Una vez abierto VS Code, ve a la pestaña de Extensiones (el icono de los cubos en la barra lateral) y busca e instala las siguientes extensiones del desarrollador Microsoft:

   - **Python:** Soporte para el lenguaje Python (linting, debugging, etc.).
   - **Jupyter:** Soporte para trabajar con Jupyter Notebooks.

Notar que al instalar las extensiones anteriores, VSCode instalará automáticamente muchas otras asociadas. En la siguiente diapositiva se muestra cómo instalar las extensiones paso a paso ↓


AutoTestIA
EXTENSIONS: MARKETPLACE
python
2
View Image for Python Debug...
simply view the image of the ima...
Elazar Cohen
10
Python Debugger
66ms
Python Debugger extension usin...
Microsoft
isort
30ms
Import organization support for P...
Microsoft
Python
124ms
Python language support with ext...
Microsoft
Pylance
322ms
A performant, feature-rich langua...
Microsoft
Python Environments
34ms
Provides a unified python environ...
Microsoft
Python Indent
15.7M
4
Kevin Rose
Install
3
Python Extension...
12.5M
4.5
Popular Visual Studio Code exten...
Don Jayamanne
Install
Python Environme...
12.5M
4
View and manage Python environ...
Welcome
main.py
main.py > ...
You, 4 days ago | 1 author (You)
import argparse
import os
import sys
from dotenv import load_dotenv
import logging
import random
import shutil
# Load .env file BEFORE importing config or pipeline
# This ensures environment variables are set when config is loaded
load_dotenv()
# Now import local modules
from autotestia.pipeline import AutoTestIAPipeline
from autotestia import config # Import config AFTER dotenv load
from autotestia import artifacts # Import artifacts module
from autotestia.artifacts import QuestionStage, ChangeMetrics # Import QuestionStage and ChangeMetrics
from autotestia.split import handle_split_command
from autotestia.merge import handle_merge_command
from autotestia.correct import handle_correct_command
from autotestia.shuffle import handle_shuffle_command
def setup_logging(log_level):
"""Configures the root logger."""
log_level_name = log_level.upper()
logging.basicConfig(level=log_level_name,
format='%(asctime)s - %
(levelname)s - %(name)s - %
CHAT
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
main.py
Add context (#), extensions (@), commanc
Agent
Auto
main
Launchpad
0 0
Live Share
R: (not attached)
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Python
Python 3.13 (64-bit)

Usando Jupyter Notebooks en VS Code

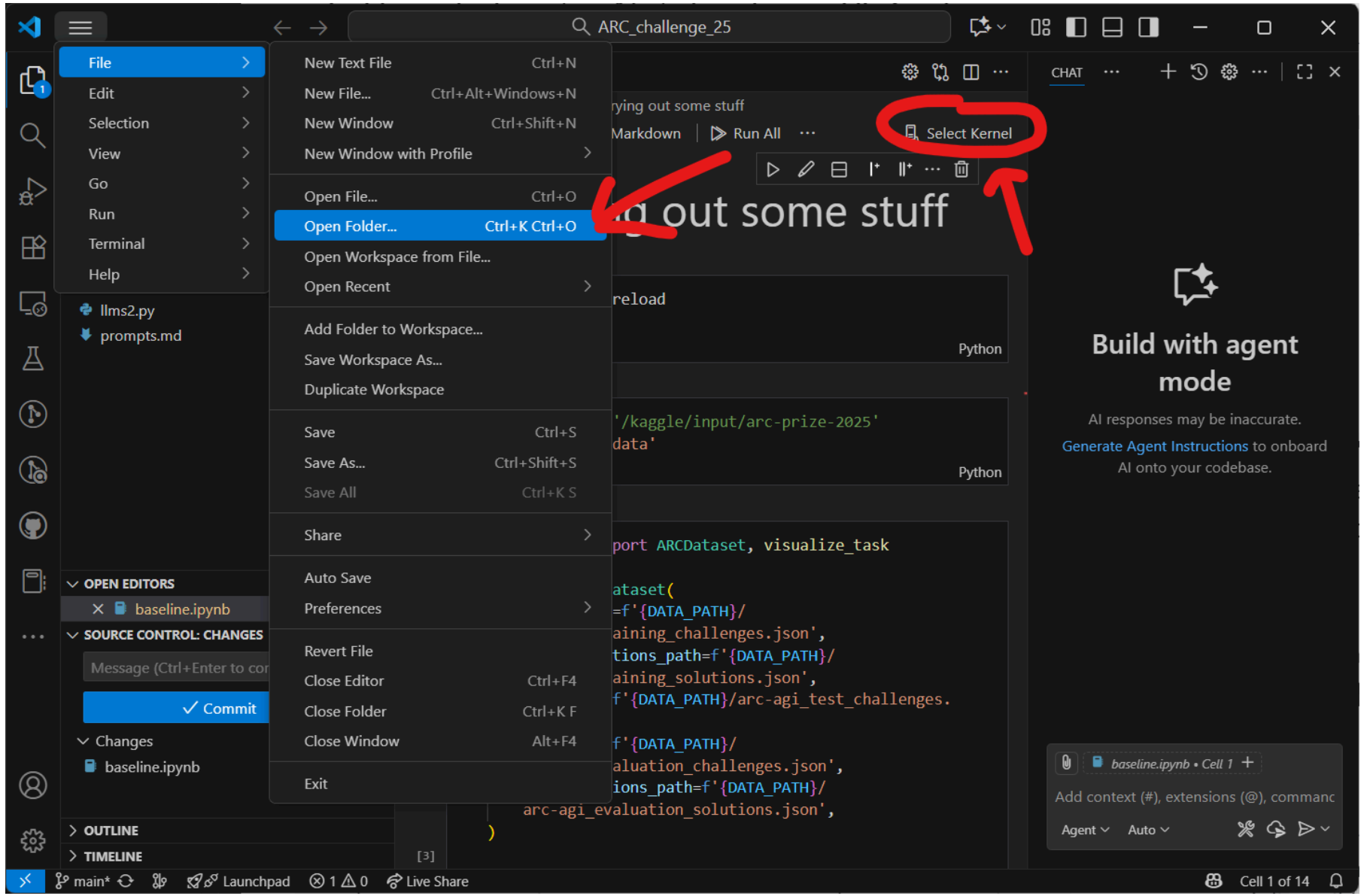
1. **Abre la carpeta de tu proyecto en VS Code**
   Puedes usar `File > Open Folder...` o arrastrar la carpeta directamente a la ventana de VS Code.

2. **Crea o abre un Jupyter Notebook**

   - Para crear uno nuevo, puedes usar el atajo `Ctrl+Shift+P` para abrir la paleta de comandos, escribir `Jupyter: Create New Jupyter Notebook` y presionar Enter.
   - Si ya tienes un archivo `.ipynb`, simplemente haz doble clic sobre él en el explorador de archivos de VS Code.

3. **Selecciona el kernel correcto**
   En la esquina superior derecha del notebook, verás un botón para seleccionar el `kernel`. Haz clic en él y elige tu entorno de conda `ds`. Si no aparece, puede que tengas que buscarlo haciendo clic en `Select Another Kernel...` y navegando hasta el ejecutable de Python de tu entorno.

Mira la imagen en la siguiente slide como referencia ↓


ARC_challenge_25
File
Edit
Selection
View
Go
Run
Terminal
Help
New Text File Ctrl+N
New File... Ctrl+Alt+Windows+N
New Window Ctrl+Shift+N
New Window with Profile
Open File... Ctrl+O
Open Folder... Ctrl+K Ctrl+O
Open Workspace from File...
Open Recent
Add Folder to Workspace...
Save Workspace As...
Duplicate Workspace
Save Ctrl+S
Save As... Ctrl+Shift+S
Save All Ctrl+K S
Share
Auto Save
Preferences
Revert File
Close Editor Ctrl+F4
Close Folder Ctrl+K F
Close Window Alt+F4
Exit
Select Kernel
Run All
Markdown
CHAT
Build with agent mode
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
llms2.py
prompts.md
OPEN EDITORS
baseline.ipynb
SOURCE CONTROL: CHANGES
Commit
Changes
OUTLINE
TIMELINE
Python
'/kaggle/input/arc-prize-2025'
data'
port ARCDataset, visualize_task
ataset(
=f'{DATA_PATH}/
aining_challenges.json',
tions_path=f'{DATA_PATH}/
aining_solutions.json',
f'{DATA_PATH}/arc-agi_test_challenges.
f'{DATA_PATH}/
aluation_challenges.json',
ions_path=f'{DATA_PATH}/
arc-agi_evaluation_solutions.json',
Add context (#), extensions (@), commanc
Agent
Auto
main*
Launchpad
Live Share
Cell 1 of 14

Accediendo a la versión Pro de GitHub Copilot

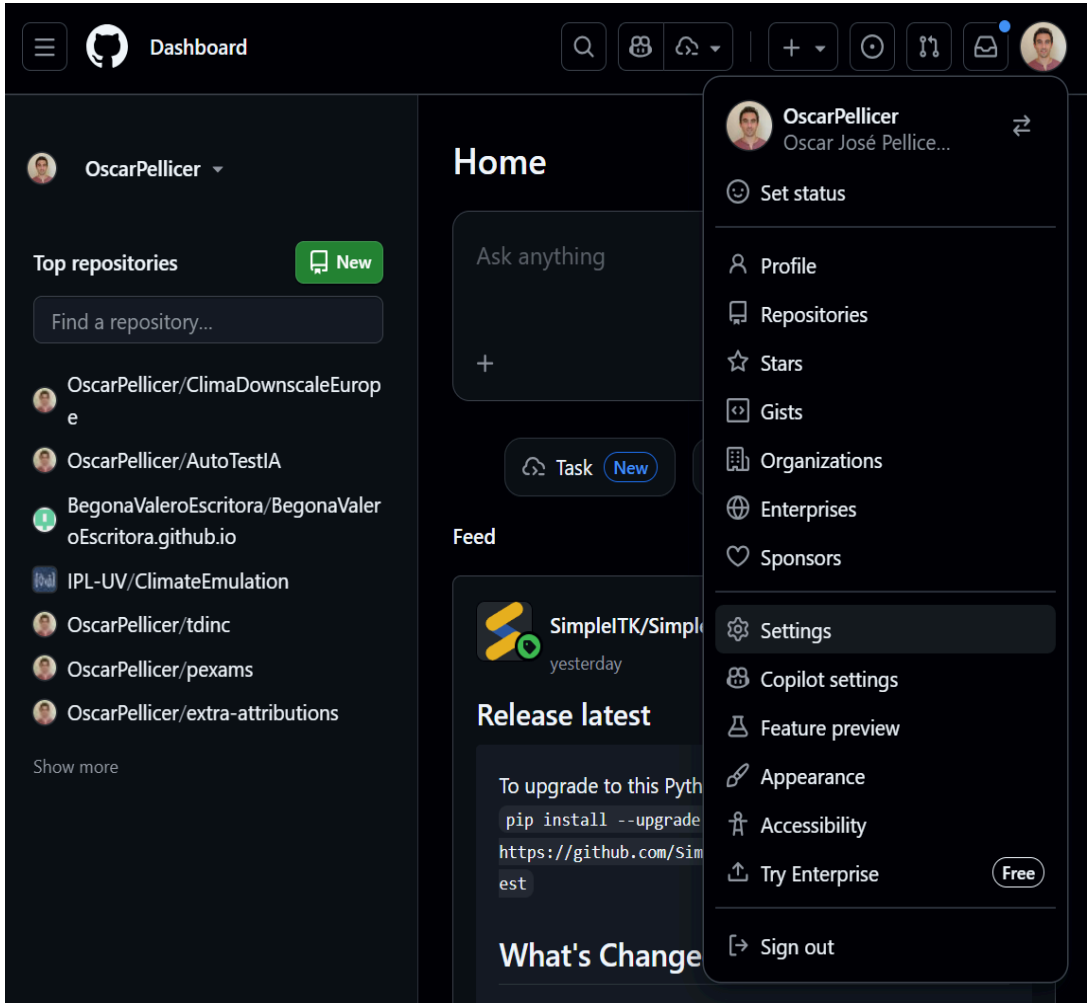
GitHub Copilot es un asistente de programación basado en IA que se integra en VS Code y te ayuda a escribir código más rápido. Como estudiantes, podéis acceder a Copilot Pro de forma gratuita solicitando a Github acceso los *Education Benefits*.

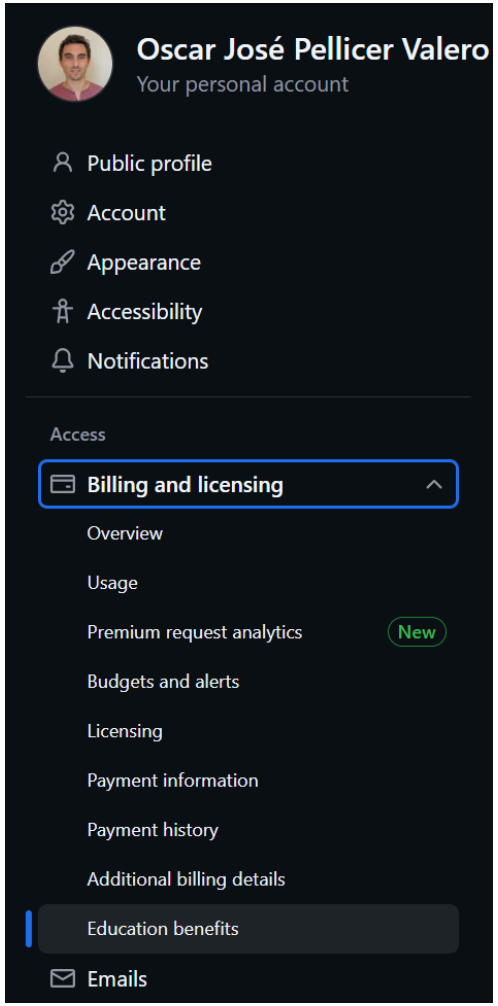
1. **Regístrate en GitHub y añade tu correo académico:**
   Si no tienes una cuenta, regístrate en GitHub usando tu correo electrónico personal (gmail, yahoo, etc.). A continuación, añade tu correo académico a tu cuenta desde la sección Emails en *Settings* (ver siguiente slide ↓)

2. **Solicita acceso a los *Education Benefits*:**
   Ve a esta página y sigue los pasos para solicitar acceso a los *Education Benefits*. Una vez aprobados, podrás acceder a Copilot Pro de forma gratuita. Nota que para ello necesitarás:

   - Subir una foto de tu carnet de estudiante (puedes usar la imagen que aparece en la intranet)
   - Tener activado la autenticación en dos factores en tu cuenta de GitHub (puedes usar una aplicación de autenticación como Microsoft Authenticator)
   - Tener tu nombre y apellidos completos en tu perfil de GitHub

3. **Instala la extensión en VS Code:**
   Ve a la pestaña de Extensiones en VS Code, busca `Github` y `GitHub Copilot` e instálalas.


Dashboard
OscarPellicer
Top repositories
New
Find a repository...
OscarPellicer/ClimaDownscaleEurope
OscarPellicer/AutoTestIA
BegonaValeroEscritora/BegonaValeroEscritora.github.io
IPL-UV/ClimateEmulation
OscarPellicer/tdinc
OscarPellicer/pexams
OscarPellicer/extra-attributions
Show more
Home
Ask anything
Task
New
Feed
SimpleITK/Simpl
yesterday
Release latest
To upgrade to this Pyth
pip install --upgrade
https://github.com/Sim
est
What's Change
OscarPellicer
Oscar José Pellice...
Set status
Profile
Repositories
Stars
Gists
Organizations
Enterprises
Sponsors
Settings
Copilot settings
Feature preview
Appearance
Accessibility
Try Enterprise
Free
Sign out


Oscar José Pellicer Valero
Your personal account
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and licensing
Overview
Usage
Premium request analytics
New
Budgets and alerts
Licensing
Payment information
Payment history
Additional billing details
Education benefits
Emails

Creando un entorno para Ciencia de Datos

Vamos a crear un entorno llamado `ds` con las librerías esenciales.

1. **Abre tu terminal (macOS/Linux) o Anaconda Prompt (Windows).**
2. **Crea el entorno instalando los paquetes directamente:**

```
conda create -n ds python=3.11 numpy pandas matplotlib scikit-learn ipykernel
```

   - `conda create -n ds`: Crea un nuevo entorno llamado `ds`.
   - `python=3.11`: Especifica la versión de Python.
   - `numpy pandas ...`: Lista de paquetes a instalar.

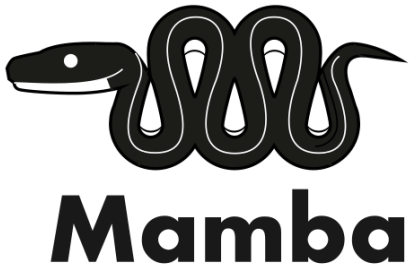
3. **Activa el nuevo entorno:**

```
conda activate ds
```

   Tu terminal cambiará para mostrar `(ds)`, indicando que el entorno está activo.

Potenciando `conda` con `mamba`

Mamba es un reimplementación de `conda` en C++ que es mucho más rápida, especialmente a la hora de resolver dependencias complejas.



Una vez instalado Miniconda, abre el **Anaconda Prompt (Windows)** (ver imagen de la derecha →) o tu **terminal (macOS/Linux)** y ejecuta:

```
conda install -n base -c conda-forge mamba
```

A partir de ahora, para la mayoría de los comandos, puedes usar `mamba` en lugar de `conda` para una experiencia mucho más fluida y rápida.

Comandos básicos de `conda` / `mamba`

Gestión de entornos:

- Crear un entorno:

```
conda create --name mi_entorno python=3.10
```

- Activar un entorno:

```
conda activate mi_entorno
```

- Desactivar el entorno actual:

```
conda deactivate
```

- Listar todos los entornos:

```
conda env list
```

- Eliminar un entorno:

```
conda env remove --name mi_entorno
```

Gestión de paquetes:

- Instalar paquetes:

```
mamba install numpy pandas scikit-learn
```

- Instalar desde `conda-forge` (más paquetes):

```
mamba install -c conda-forge nombre_paquete
```

- Listar paquetes instalados en el entorno activo:

```
mamba list
```

- Eliminar un paquete:

```
mamba remove nombre_paquete
```

- Exportar un entorno a un archivo y crearlo desde él:

```
mamba env export > entorno.yml
mamba env create -f entorno.yml
```

Alternativa a VS Code: Jupyter Lab

Si prefieres la interfaz clásica de Jupyter Lab en el navegador, puedes seguir estos pasos.

1. **Instala Jupyter Lab en el entorno `base`:** Jupyter Lab es una interfaz web interactiva. Lo instalaremos en el entorno `base` para que actúe como nuestro "lanzador" principal.

   ```
   mamba install -n base -c conda-forge jupyterlab
   ```

2. **Conecta tu entorno `ds`:** Por defecto, Jupyter Lab solo "ve" el Python del entorno desde el que se lanza. Para que pueda usar nuestro entorno `ds`, necesitamos instalar `ipykernel` en él y registrarlo.

   - **Activa el entorno `ds`:** `conda activate ds`
   - **Instala `ipykernel`:** `mamba install ipykernel`
   - **Registra el entorno:** `python -m ipykernel install --user --name ds --display-name ds`

3. **Lanza Jupyter Lab:**
   Asegúrate de estar en el entorno `base` (`conda deactivate` si estás en otro) y ejecuta: `jupyter lab`
   Esto abrirá Jupyter Lab en tu navegador, y ahora verás "ds" como una opción para crear nuevos notebooks.

Alternativa a `conda` / `mamba`: `venv`

Python incluye su propio módulo para crear entornos virtuales ligeros, llamado `venv`.

Creación y activación:

1. **Navega a la carpeta de tu proyecto.**
2. **Crea el entorno:**

   ```
   python -m venv .venv
   ```

   (Se creará una carpeta `.venv` en tu directorio actual)
3. **Activa el entorno:**
   - **Windows:**

     ```
     .venv\Scripts\activate
     ```

   - **macOS / Linux:**

     ```
     source .venv/bin/activate
     ```

Gestión de paquetes:

- `venv` no gestiona paquetes. Se usa `pip`, el instalador de paquetes de Python.
- **Instalar paquetes:**

  ```
  pip install numpy pandas
  ```

- **Guardar dependencias:**

  ```
  pip freeze > requirements.txt
  ```

- **Instalar desde un archivo:**

  ```
  pip install -r requirements.txt
  ```

`conda` / `mamba` vs. `venv`: Diferencias clave

`conda` / `mamba`

- **Gestor de paquetes Y de entornos.**
- **Independiente del lenguaje:** Puede gestionar paquetes y dependencias de otros lenguajes (R, C++, etc.), no solo Python.
- **Gestión de binarios:** Instala paquetes pre-compilados (binarios), lo que puede ser más rápido y evitar problemas de compilación, especialmente en Windows.
- **Gestión de Python:** Puede instalar diferentes versiones de Python en diferentes entornos.
- **Centralizado:** Los entornos se guardan en una carpeta central de Miniconda, no junto al proyecto.

`venv`

- **Solo gestor de entornos.** Para paquetes, se usa `pip`.
- **Específico de Python:** Solo gestiona paquetes de Python.
- **Instala desde "wheels" o fuentes:** `pip` instala paquetes binarios (wheels) o compila desde el código fuente si no hay un wheel disponible para tu sistema.
- **Usa el Python con el que se creó:** El entorno usa la versión de Python del sistema que se usó para crearlo.
- **Descentralizado:** El entorno se crea típicamente dentro de la carpeta del proyecto.

Recomendación: Para este curso, usaremos **`conda`** + **`mamba`** por su robustez en el ecosistema de ciencia de datos.