Práctica 0. Python

Objetivo

El objetivo de esta práctica es simplemente que os familiaricéis con Python y con las librerías que principalmente vamos a utilizar en las prácticas posteriores. La práctica 0 no se evalúa.

Python

Python es un lenguaje de programación de alto nivel, multi-paradigma y de propósito general. Es interpretado, tiene tipado dinámico y cuenta con un manejo de memoria automático.

Estas características, junto con el amplio repertorio de librerías, como p. ej. **NumPy** (para agilizar cálculos numéricos) y **scikit-learn** (para el diseño e implementación de algoritmos de aprendizaje automático), hacen que este lenguaje sea especialmente útil para el prototipado y comprensión de las técnicas que se tratarán en la asignatura.

Flujos de trabajo

Para facilitar la instalación de Python y el manejo de dependencias, recomendamos:

- Si se trabaja en *local* (con el hardware personal): instalar **Anaconda** o **Miniconda**, ya que incluyen el administrador de paquetes **conda**, el cual facilita enormemente el manejo e instalación de librerías mediante *entornos virtuales* (distribuciones aisladas de Python). Estos entornos virtuales pueden emplearse fácilmente en IDEs populares (como **Spyder** o **Visual Studio Code**), así como en *notebooks* interactivos de **Jupyter** (los cuales se ejecutan en nuestro navegador web).
- Si se prefiere trabajar en *remoto*: emplear **Google Colaboraty**. Google Colab es una plataforma interactiva que corre enteramente en la nube. Tiene pre-instaladas la práctica totalidad de las librerías que emplearemos. Es necesario tener una cuenta de Google Drive (en la que se guarda nuestro trabajo) y que además podemos "montar" para importar y editar módulos (archivos de Python) que tengamos almacenados en ella.

Existen múltiples formas de trabajar con estas herramientas y su documentación online es vasta, por lo que sentiros libres de trabajar de la manera que os resulte más cómoda. Por si acaso resulta de utilidad, a continuación indicamos brevemente cómo podemos trabajar con Miniconda.

Trabajando en local con Miniconda

Miniconda **cuenta con instrucciones** de instalación en Windows, macOS y Linux. Tras instalar Miniconda, podemos crear ya el entorno virtual que emplearemos en las prácticas.

> **Warning**
>
> Anaconda **desaconseja** instalar librerías en el entorno `(base)`. Por esta, entre otras razones, lo más recomendable es crear un entorno virtual para la asignatura.

Este entorno lo podemos crear de forma manual desde la terminal (en macOS y Linux) o "Anaconda Prompt" (en Windows). Tal y como se indica en la diapositiva 30 de introducción de la asignatura, ejecutando

```
>_
(base) cualquier/path>conda create -n apr24 python=3.10
```

e introduciendo `y` a la pregunta `Proceed ([y]/n)?`, habremos creado un entorno virtual llamado `apr24` e instalado `python v3.10` en él (además de otras librerías básicas que por defecto se instalan siempre, como p. ej. `pip`).

Siempre que queramos trabajar con este entorno, deberemos activarlo si no lo está ya. Es decir, en la terminal, en vez de `(base)` (o cualquier otro entorno que hayamos creado) debe aparecer `(apr24)`. Esto se consigue con `conda activate`:

```
>_
(base) cualquier/path>conda activate apr24
# Tras ejecutarlo, aparecerá (apr24), indicando que está activo:
(apr24) cualquier/path>
```

Así, ya podemos instalar en este entorno librerías fundamentales para las prácticas:

```
>_
(apr24) cualquier/path>conda install scikit-learn matplotlib pandas seaborn
```

De nuevo, deberemos confirmar su instalación con `y`.

> **Info**
>
> A pesar de no haber indicado con el comando anterior la instalación de librerías relevantes como `Numpy` o `Scipy`, éstas también se han instalado y podemos trabajar con ellas ya que son dependencias de las librerías que sí hemos indicado.

Si más adelante se quieren instalar nuevas librerías, lo podemos hacer fácilmente y de igual forma a como lo hemos hecho antes. Por ejemplo, `Keras` es una librería relevante que podemos emplear en las prácticas. Podemos instalarla simplemente ejecutando:

```
>_
(apr24) cualquier/path>conda install keras
```

`conda` se encargará de resolver las dependencias e instalar una versión adecuada.

> **Tip**
>
> En Windows, podemos incorporar fácilmente *Anaconda Prompt* a la **Terminal de Windows** añadiendo el objeto correspondiente a *Anaconda Prompt* en la lista de perfiles (`profiles`) en el `settings.json` de la terminal. Asumiendo que se ha instalado miniconda en el directorio por defecto, dicho objeto debería ser similar a:

```
>_

{
    "commandline": "cmd.exe /K %USERPROFILE%\\miniconda3\\Scripts\\activate.bat",
    "guid": "{055e4a8f-f691-4a10-a651-ed5721580733}",
    "hidden": false,
    "icon": "%USERPROFILE%\\miniconda3\\Menu\\Iconleak-Atrous-Console.ico",
    "name": "Anaconda-Prompt-(miniconda3)",
},
```

`"guid"`s diferentes, pueden generarse, p. ej., en https://guidgenerator.com/.

Uso de IDEs con Miniconda

Para trabajar con un entorno virtual de Miniconda (como `(apr24)`) en IDEs populares como **Spyder** o **VSCode**, simplemente debemos lanzarlas desde la terminal (Anaconda Prompt en Windows) con el entorno virtual de interés activado:

```
>_

# lanzar spyder en el directorio path/to/project:
(apr24) cualquier/path>spyder -p path/to/project

# lanzar VSCode en el directorio path/to/project:
(apr24) cualquier/path>code path/to/project
```

En cuanto a la instalación de las IDEs, con VSCode no hay problema si la instalamos desde su **página web**. Sin embargo, en el caso de Spyder, recomendamos instalarla con `conda`, para así evitar **configuraciones adicionales necesarias** o **problemas en Linux**:

```
>_

(apr24) cualquier/path>conda install spyder
```

Jupyter Notebooks

Los **Jupyter noteboks** son también una herramienta muy popular para desarrollar proyectos en Python. Permiten prototipar y testear *celdas* (piezas) de código de forma muy sencilla, así como añadir explicaciones del mismo con celdas de texto e imágenes. **Google Colab** puede entenderse como una versión en la nube de los Jupyter notebooks.

Su versión más simple puede instalarse con conda:

```
>_

(apr24) cualquier/path>conda install jupyter
# para lanzarlo:
jupyter notebook /path/to/project
```

Alternativamente, tras instalar `jupyter`, en vez de usar `jupyter notebook` para lanzar los notebooks, podemos emplear la misma **funcionalidad ya integrada en VSCode**.

En la siguiente sección, os recomendamos un tutorial que sirve como práctica de tanto Jupyter Notebook, como Google Colab.

Recursos adicionales

Si es la primera vez que se trabaja con Python, además de los recursos indicados en la diapositiva 32 de introducción de la asignatura, recomendamos seguir el tutorial: **Python Numpy Tutorial (with Jupyter and Colab)** por ser conciso y tener un enfoque práctico en los usos más habituales de Python y Numpy, además de servir de práctica de tanto Jupyter notebooks como Google Colab.

Durante las prácticas vais a utilizar, además de Numpy, de forma habitual estas dos librerías con las que os recomendamos familiarizaros:

- **Pandas** para la carga y manipulación de datos. La estructura de datos principal es el dataframe. Un DataFrame es una estructura de datos bidimensional donde los datos se alinean de forma tabular en filas y columnas. Un Dataframe consta de tres componentes principales, los datos, las filas y las columnas. Os recomendamos aprender a crear, manipular y acceder a los dataframes ya que los usaréis tanto para cargar datos como para almacenar resultados. Tenéis más informacion en este **tutorial de Pandas**.
- **Matplotlib** para la visualización de resultados. Os permite crear figuras, manipularlas y exportarlas. Además de la página de referencia, aquí tenéis un un buen **tutorial práctico**. En este caso os recomendamos aprender a utilizar la librería para crear figuras.