

Big Data GIS (GeoAnalytics) y Análisis Espacial

Documentación oficial Esri España

PARTE - 2

ÍNDICE

1. ARCGIS NOTEBOOKS	4
¿Qué es ArcGIS Notebooks?	4
ArcGIS Notebooks (ArcGIS Enterprise)	4
ArcGIS Notebooks (ArcGIS Online)	5
ArcGIS Notebooks (ArcGIS Pro)	6
Integración ArcPy y Python API	7
Acceso a ArcGIS Notebooks	7
Gestión de ArcGIS Notebooks	9
EJERCICIO 1: ARCGIS NOTEBOOKS	10
Paso 1: Accede a ArcGIS Online	10
Paso 2: Creando notebooks	10
Paso 3: Importar el paquete pandas	12
Paso 4: Markdown	13
Paso 5: Uso de la cinta de ArcGIS Notebooks en ArcGIS Online	16
2. HERRAMIENTAS PYTHON DE UTILIDAD	18
Python para la ingeniería de datos	18
EJERCICIO 2: CONOCIENDO LAS LIBRERÍAS IMPORTANTES DEL USO DE API DE PYTHON	21
Paso 1: Comenzando con NumPy	21
Paso 2: Comenzando con Pandas	23
Paso 3: Comenzando con Matplotlib	31
EJERCICIO 3: MATPLOTLIB (PAMPLONA)	36
Paso 1: Representación básica con plot	36
Paso 2: Eliminar campo innecesario	37
Paso 3: Aplicar el método set_xticklabels()	37
Paso 4: Filtrar registros por año	39
3. INTRODUCCIÓN A LA API DE PYTHON	42
Arquitectura	42
Módulos principales	44
Spatially enabled DataFrame	45
El widget MapView	45
Módulo GIS	46

Patrones en el módulo GIS y en la API	48
EJERCICIO 4: GESTIÓN DE CONTENIDOS Y USUARIOS	49
Paso 1: Buscar elementos	49
Paso 2: Resumen de la actividad de un usuario	52
4. PERFILES DE USUARIO	54
Administración de tu Web GIS	55
Generando contenidos	55
EJERCICIO 5A: GENERAR UNA VISUALIZACIÓN EN ARCGIS NOTEBOOKS	56
Paso 1: Analizar los datos mediante diferentes librerías	56
Paso 2: Crear un widget mapa para visualización	59
Paso 3: Actualizar propiedades del mapa	59
Paso 4: Aplicar la simbología smart mapping	61
EJERCICIO 5B: SMART MAPPING CON PERSONALIZACIÓN DEL RENDERIZADOR	64
Paso 1: Trabajar con el renderizador básico de Smart mapping	64
Paso 2: Personalizar el renderizador	66
5. MÓDULO FEATURES	70
EJERCICIO 6: CONSULTANDO DATOS	72
Paso 1: Acceso al portal GIS	72
Paso 2: Consulta de los datos	73
Paso 3: Mapa que muestre los resultados de la consulta	76
EJERCICIO 7: ANÁLISIS DE LOS AVISOS RECIBIDOS EN EL DISTRITO DE MORATALAZ	82
Paso 1: Abrir el proyecto y crear un nuevo Notebook	82
Paso 2: Buscar la capa Moratalaz Avisos dentro del contenido del portal	82
Paso 3: Crear variables que guarden los avisos y los límites de los barrios	84
Paso 4: Agregación de los avisos por barrios	85
Paso 5: Cambiar la simbología del mapa	87
Paso 6: Consultar los datos del análisis de agregación	89
Paso 7: Visualizar la distribución del número de avisos por barrio en una grafica	90
6. INTRODUCCIÓN AL ANÁLISIS ESPACIAL CON ARCGIS API FOR PYTHON	93

Análisis de redes (Network Analysis)	93
Análisis de imágenes y ráster (Image & Raster Analytics)	94
Ventajas del análisis espacial con ArcGIS API for Python	95

EJERCICIO 8: ANÁLISIS ESPACIAL CON REDES 96

Paso 1: Conexión al portal y librerías	96
Paso 2: Cargar los datos desde el CSV	96
Paso 3: Crear un mapa y visualizar los puntos	97
Paso 4: Trabajar con el módulo arcgis.network	98
Paso 5: Ejecutar el cálculo de áreas de servicio (1, 3 y 5 minutos)	99
Paso 6: Preparar el resultado para representarlo en el mapa	100
Paso 7: Definir la simbología según el tiempo (1, 3, 5)	101
Paso 8: Añadir las áreas de servicio al mapa con su simbología	102
Interpretación del resultado	104

EJERCICIO 9: ANÁLISIS DE IMÁGENES Y ANÁLISIS RÁSTER 105

Paso 1: Acceder a las imágenes directamente con ArcGIS Living Atlas	105
Paso 2: Acceso a datos Sentinel del Living Atlas	107
Paso 3: Trabajar con funciones ráster	108
Paso 4: Guardar imágenes	110

1. ArcGIS Notebooks

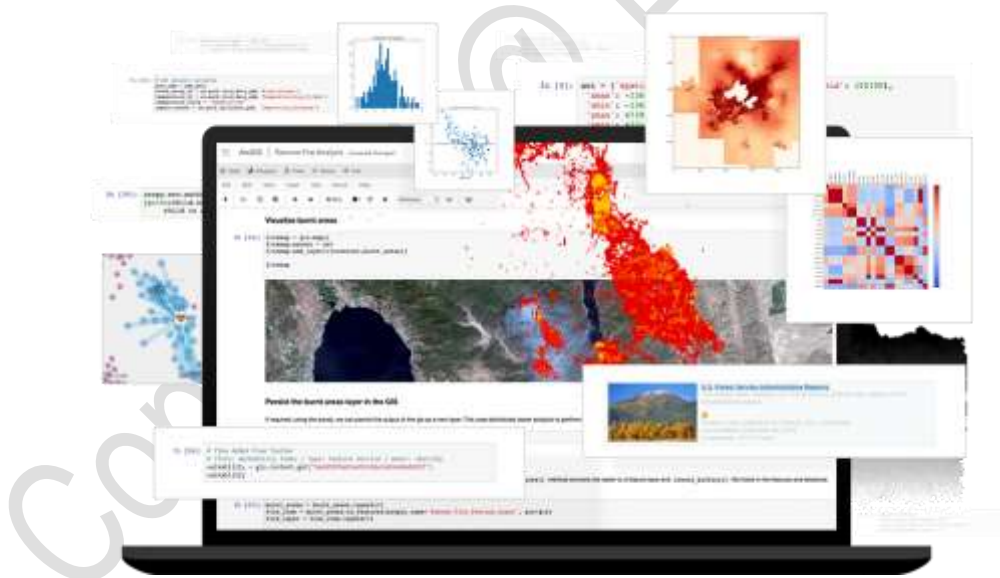
¿Qué es ArcGIS Notebooks?

ArcGIS Notebooks ofrece una versátil interfaz web para realizar potentes análisis de datos geoespaciales. Con los notebooks, usted puede realizar análisis, automatizar flujos de trabajo y ver al momento los resultados de los datos y los análisis en un contexto geográfico.

Los notebooks son un entorno eficiente y moderno que combina código, visualizaciones y mapas al vuelo y herramientas de datos. En el editor de notebooks, puede escribir, documentar y ejecutar código de Python en un único lugar.

ArcGIS Notebooks ofrece una plataforma integrada con la que crear, compartir y ejecutar ciencia de datos, administración de datos y secuencias de comandos administrativas. Los autores de notebooks pueden utilizar los recursos de Python de Esri (por ejemplo, ArcGIS API for Python y ArcPy), así como populares bibliotecas de código abierto analíticas, estadísticas y de aprendizaje automático.

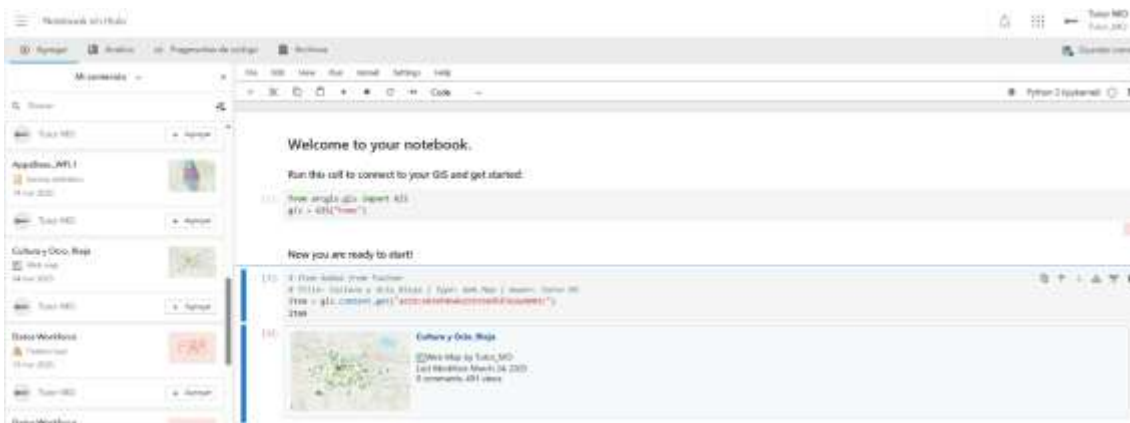
Python es una parte importante de ArcGIS y con ArcGIS Notebooks puede incorporar la versatilidad de Python directamente en el portal de ArcGIS Enterprise. La API le permite incorporar mapas dinámicos y herramientas de datos geoespaciales en sus notebooks.



ArcGIS Notebooks (ArcGIS Enterprise)

A partir de la versión 10.7 de ArcGIS Enterprise, contamos con un rol que habilita el uso de Notebooks en tu portal.

La integración de Notebooks en su portal hace más rápido del prototipado de scripts, pudiendo cargar directamente datos y herramientas GIS en una celda.



Por defecto, hay unas librerías base que puede utilizar en estos notebooks, aunque es posible que quiere agregar alguna más.

Todos los ArcGIS Notebooks se ejecutan en ArcGIS Server con licencia y el rol ArcGIS Notebook Server. Una vez que este servidor esté federado con su portal, se puede crear un notebook de la misma forma que una capa o aplicación web del sitio web del portal. Los ArcGIS Notebooks utilizan un tipo de seguridad basada en identidades como cualquier otro elemento de su portal: los administradores controlan quienes pueden crearlos, compartirlos, editarlos y verlos.

En ArcGIS Enterprise siempre la versión dependerá de la versión del producto que define el equipo de sistemas en su organización.

Runtime Version	ArcGIS Enterprise
12.0	11.5
11.0	11.4
10.0	11.3
9.0	11.2
8.0	11.1
7.0	11.0
6.0	10.9.1
5.0	10.9

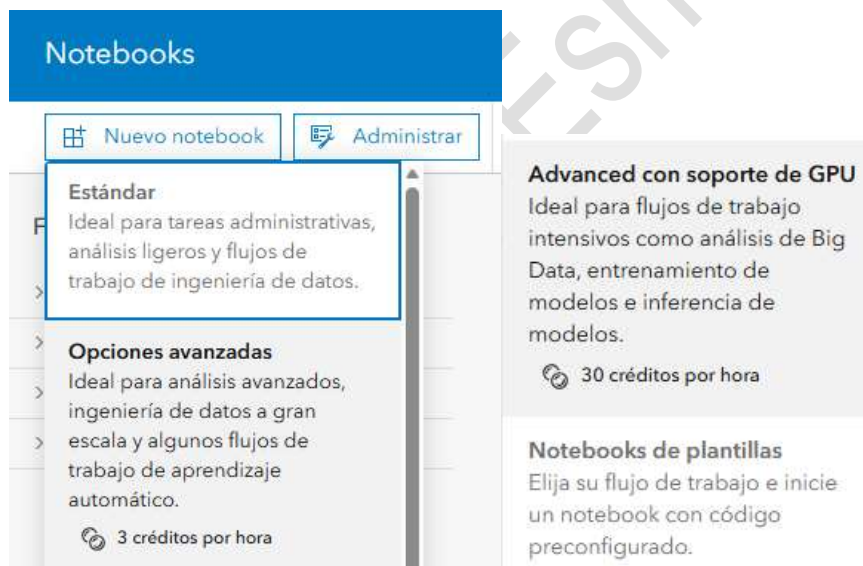
ArcGIS Notebooks (ArcGIS Online)

En ArcGIS Online, usted también tiene acceso a ArcGIS Notebooks, aunque no es posible modificar las librerías del notebook base.

Con la actualización de junio de 2020 de **ArcGIS Online**, **ArcGIS Notebooks** pasó a la disponibilidad general, conteniendo cientos de bibliotecas **Python** de código abierto junto con las bibliotecas **ArcGIS Python**, **ArcPy** y **ArcGIS API for Python**.

Existen **4 tipos** de **Notebooks**:

- **Estándar:** incluye la **ArcGIS API for Python** y cientos de bibliotecas de código abierto; apropiado para tareas administrativas, ingeniería de datos y análisis ligero. Sin coste de créditos.
- **Avanzado:** incluye todo lo que se ofrece en **Estándar** más **ArcPy**; ofrece un tamaño de instancia más grande (más recursos informáticos y de memoria); apropiado para la mayoría de los flujos de trabajo, incluida la ingeniería de datos a gran escala, el análisis avanzado y algunos flujos de trabajo de aprendizaje automático. Coste de 3 créditos/hora.
- **Avanzado con GPU:** incluye todo en **Estándar** y **Avanzado** junto con una instancia impulsada por GPU que ofrece la mayor cantidad de recursos informáticos y de memoria; apropiado para flujos de trabajo computacionalmente intensivos, como análisis de big data, entrenamiento de modelos e inferencia de modelos. Coste de 30 créditos/hora.
- **Notebooks de plantillas:** una experiencia de bajo código que permite a los usuarios seleccionar entre una colección de notebooks listos para usar organizados en categorías predefinidas que les permiten comenzar con un ejemplo práctico.

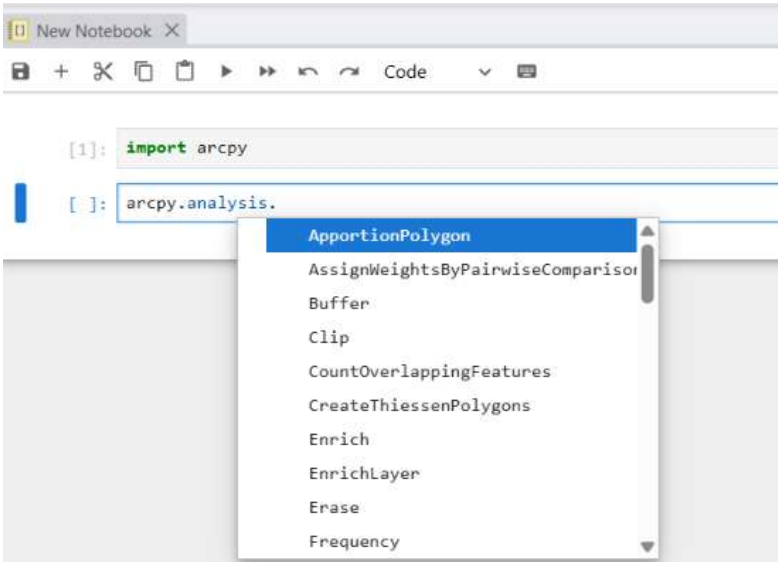


En ArcGIS Online siempre contaremos con la última versión desarrollada por Esri.

ArcGIS Notebooks (ArcGIS Pro)

ArcGIS Notebooks está basado en [Jupyter Notebook](#), una aplicación web de código abierto que le permite crear y compartir documentos que contienen código Python en vivo, visualizaciones y texto narrativo. Con ArcGIS Notebooks integrado en ArcGIS Pro, puede realizar análisis y ver resultados inmediatamente en un contexto geográfico, interactuar con los datos emergentes, documentar y automatizar su flujo de trabajo, y guardarlo para usarlo más adelante o compartirlo. Los usos de ArcGIS Notebooks incluyen limpieza y transformación de datos, simulación numérica, modelado estadístico, aprendizaje automático, tareas administrativas y más.

Toda la funcionalidad de Python en ArcGIS Pro está disponible a través de ArcGIS Notebooks , incluida la funcionalidad principal de Python , la biblioteca estándar de Python, [ArcPy](#), [ArcGIS API for Python](#) y las numerosas bibliotecas de terceros incluidas con ArcGIS Pro , como NumPy y pandas. ArcGIS Pro se puede ampliar con bibliotecas de código abierto utilizando ArcGIS Pro [Package Manager](#).



Integración ArcPy y Python API

Una opción interesante es utilizar al mismo tiempo ArcPy y el API de Python, igual que integramos diferentes paquetes en nuestros scripts.

Como ya sabe ArcPy forma parte del core de los productos de Esri: ArcGIS Pro y ArcGIS Enterprise. Por lo tanto, acceder a este paquete, significa de algún modo acceder a estos productos.

La integración del API de Python con ArcPy, solo puede darse en ciertos casos, del mismo modo que la integración con otros paquetes:

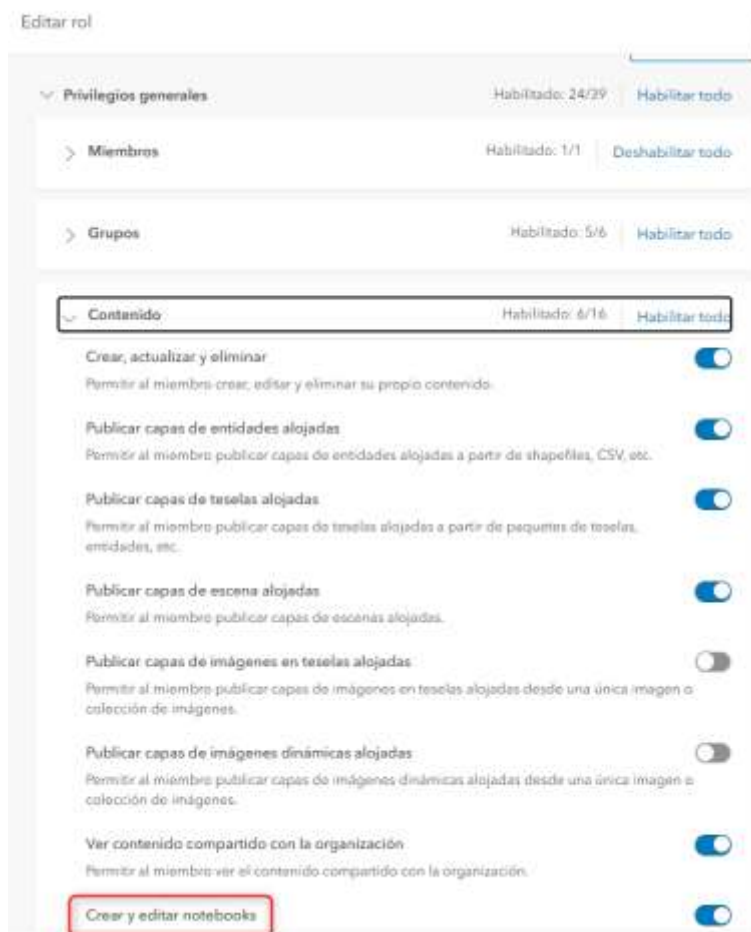
Entorno	Uso de ArcPy	Instalación de paquetes Python adicionales
ArcGIS Pro	Sí	Sí, realizando una copia del entorno por defecto
ArcGIS Enterprise	Sí, con licencia Advanced	Sí
ArcGIS Online	Sí	Temporalmente

Acceso a ArcGIS Notebooks

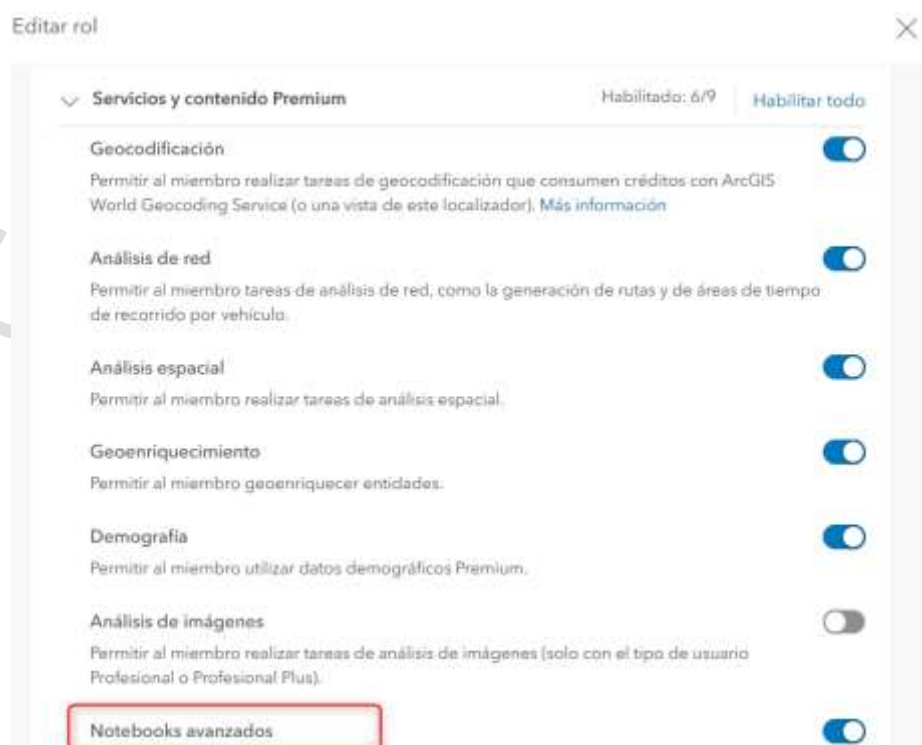
Existen roles personalizados en ArcGIS Online y ArcGIS Enterprise que requieren contar con los privilegios necesarios para acceder a ArcGIS Notebooks, que en ArcGIS Enterprise solo estarán visibles si está configurado Notebooks Server previamente.

(<https://enterprise.arcgis.com/es/notebook/latest/install/windows/install-arcgis-notebook-server.htm>)

En concreto, es necesario habilitar a nivel de “Contenido” la creación y edición de notebooks:



Adicionalmente, si quiere que los miembros de un rol tengan acceso a ArcGIS Notebooks Advanced, habilitará la opción correspondiente dentro de la sección de "Contenido Premium":

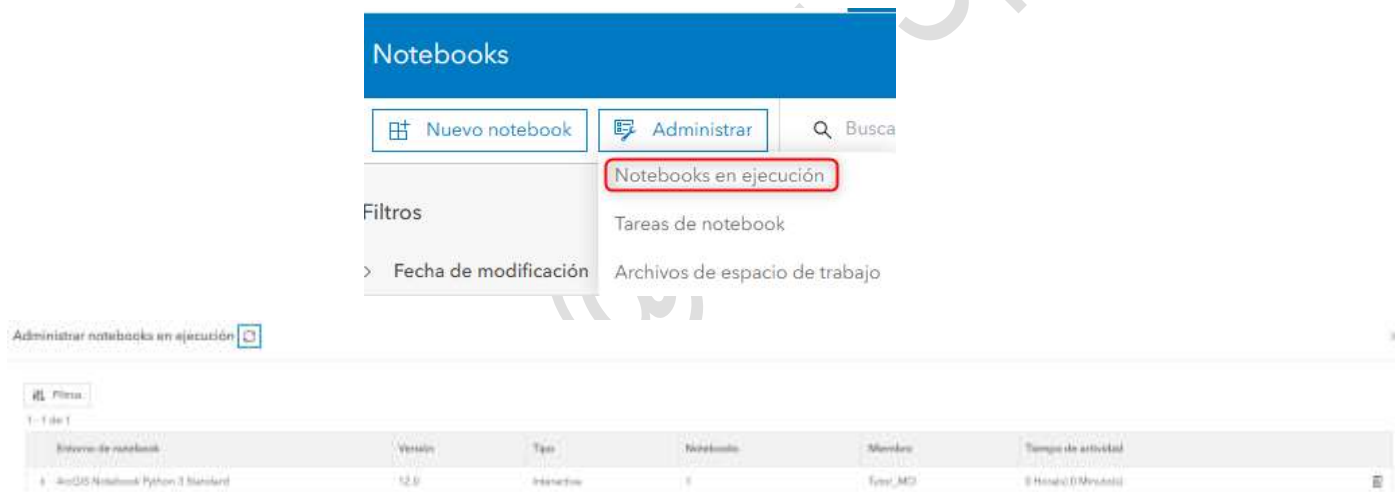


Gestión de ArcGIS Notebooks

Los notebooks son un ítem más dentro de la Plataforma ArcGIS. Como tal, puede crearlos, editarlos, buscarlos y compartirlos.



Además, como administrador, puede en todo momento descubrir los notebooks que se están ejecutando y, si fuera preciso, detenerlos: **Administrar ->Notebooks en ejecución:**



Ejercicio 1: ArcGIS Notebooks

Una vez que cuente con un entorno de los posibles (local con ArcGIS Pro o ArcGIS Online/ArcGIS Enterprise), va a realizar la creación y prueba de notebooks.

En este caso, va a utilizar ArcGIS Notebooks en ArcGIS Online, cuya mayor ventaja respecto al resto de opciones es la facilidad de uso.

Paso 1: Accede a ArcGIS Online

- Acceda a la organización de ArcGIS Online:

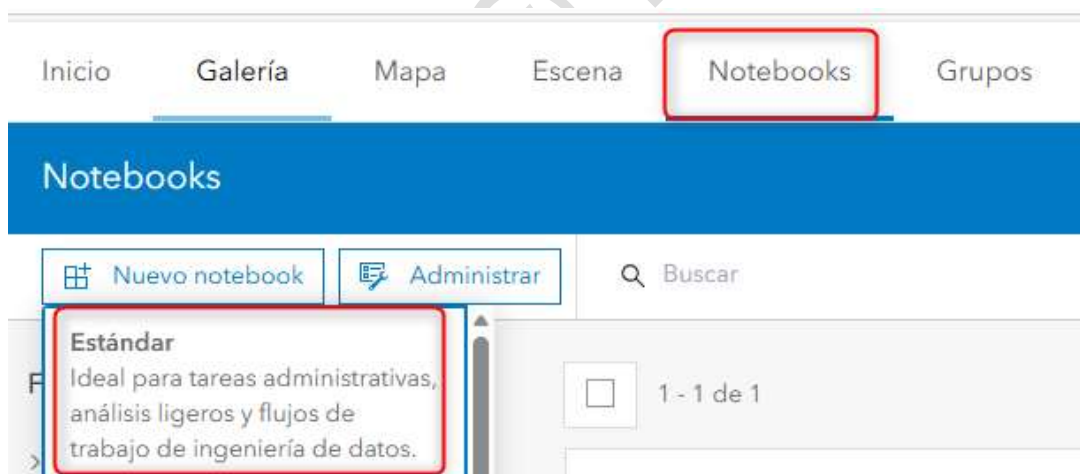


Dado que pertenece a un rol que tiene acceso a ArcGIS Notebooks, verá directamente en la barra superior un acceso a ArcGIS Notebooks.

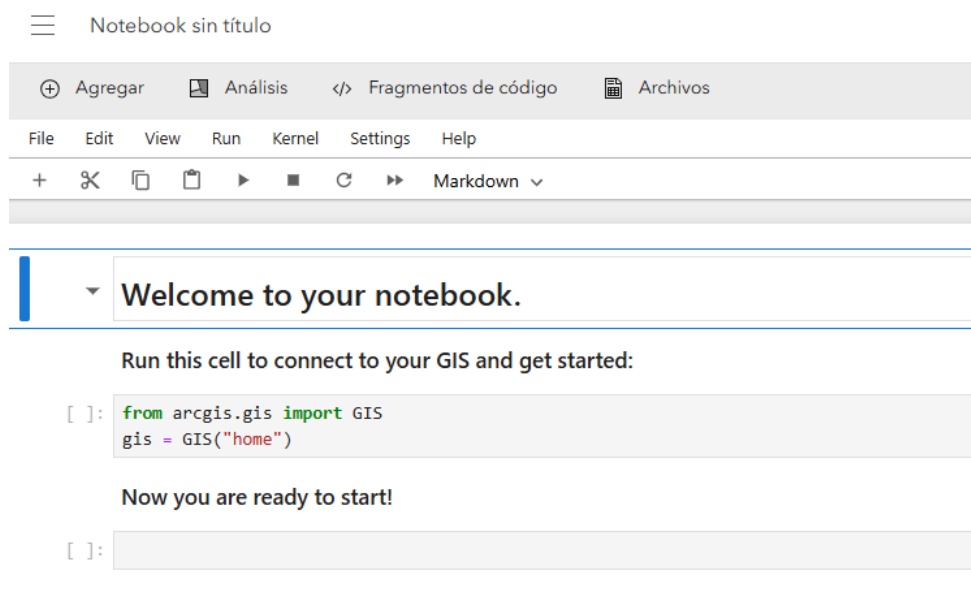
Paso 2: Creando notebooks

Ya está en disposición de crear su primer notebook.

- Haga clic en el botón de "Nuevo notebook" e indique que quiere crear un notebook de tipo "Estándar":



Una vez creado el primer notebook, ya tiene disponible un entorno Python sobre el cual va a poder trabajar para administrar su organización, gestionar sus datos, servicios y aplicaciones y realizar análisis espacial.



ArcGIS Notebooks, lógicamente al estar basado en Jupyter, se basa en el uso de celdas (cells) mediante las cuales se interactúa con el intérprete de Python (kernel).

Puede escribir su código Python en estas celdas y ejecutarlo como si de scripts aislados se tratara, aunque con la ventaja de que sus variables están disponibles de una celda a otra:

Nota: Recuerde que pueda insertar nuevas filas de código pulsando el comando de la tecla B y Shift+Enter para ejecutar la celda.

Now you are ready to start!

```
[1]: a = 4
      b = 5

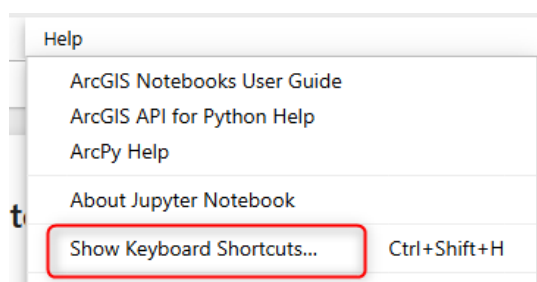
[2]: a+b

[2]: 9
```

- Ejecute la celda con el botón "Run" o con atajos de teclado.

Existen multitud de atajos de teclado que le permiten ser muy ágil.

- En la pestaña "Ayuda" busque la opción "Show Keyboard Shortcuts" para ver todos ellos.



- Una vez que esté trabajando en un notebook, puede cargar paquetes de Python (además del kernel de Python ya precargado) y utilizarlos del mismo modo que en sus scripts:

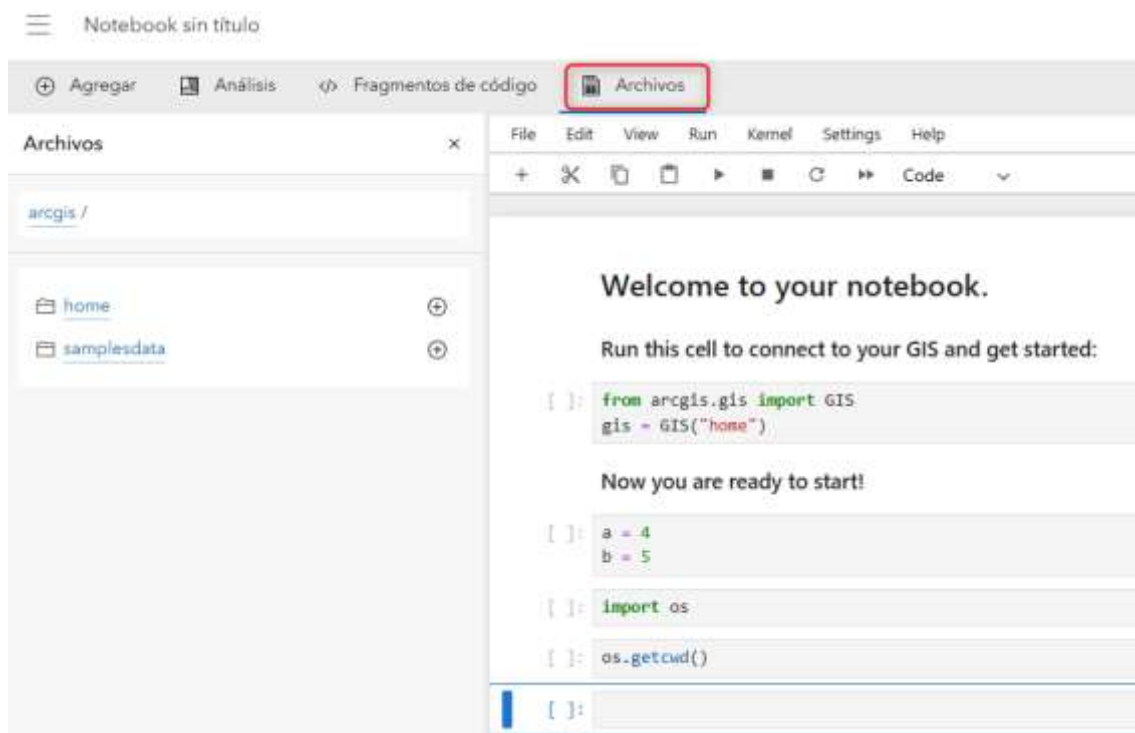
```
import os
```

```
os.getcwd()
```

```
 '/arcgis'
```

En este ejemplo, está accediendo a la ruta en la cual está trabajando en su notebook. Verá que su notebook trabaja sobre una carpeta llamada "arcgis".

Nota: ArcGIS Notebooks pone a nuestra disposición una carpeta de trabajo. Puede acceder a ella pinchando sobre el botón "Archivos" del menú superior.



Paso 3: Importar el paquete pandas

Nota: Este proceso es muy habitual para la gestión de datos tabulares (CSV, por ejemplo): <https://pandas.pydata.org/pandas-docs/stable/index.html>

- Importe el paquete pandas y úselo para leer el siguiente CSV (recuerde poner la URL entre comillas simples):

https://raw.githubusercontent.com/playgis/devsummit2017/master/python/major_earthquakes.csv

```
[1]: import pandas as pd
```

```
[2]: pd.read_csv('https://raw.githubusercontent.com/playgis/devsum12017/master/python/major_earthquakes.csv')
```

```
[7]:
```

	Unnamed: 0	X	Y	ProgramName	ProjectTitle	DamageExtent	ID
0	0	-118.213772	34.032372	Municipal Facilities R and P	Boyle Heights Sports Center	95	35
1	1	-118.245915	34.045172	Municipal Facilities R and P	Engine Company 23 Arts Center	76	20565
2	2	-118.245911	34.045161	Municipal Facilities R and P	Engine Company 23 Arts Center	76	20566
3	3	-118.407213	34.235234	Bridge Improvement Program	Laurel Canyon Blvd Bridge-1233	78	771
4	4	-118.317632	34.025554	Wastewater - CSP	Wilshire Area System Swr Rehab	58	768
5	5	-118.432091	33.928674	Wastewater - TFP	HWRP- IPS Odor Control Improvements	62	180
6	6	-118.479765	34.181202	Wastewater - TFP	DCT- Backup Power	55	285
7	7	-118.405540	34.144947	Municipal Facilities R and P	Los Angeles Riverfront Park Phase II	90	109

Uno de los beneficios de ArcGIS Notebooks es que podrá visualizar información en tiempo real tras la ejecución de cada celda. Esto le permite depurar sus scripts y, al compartir sus notebooks, quienes los reciban podrán comprender mejor su código y sus objetivos.

Este formato permite, por tanto, generar código que sea fácilmente reproducible y utilizable por otros usuarios, pudiendo ser adaptado a nuevos datos, configuraciones, etc. sin necesidad de reescribir complicadas funciones.

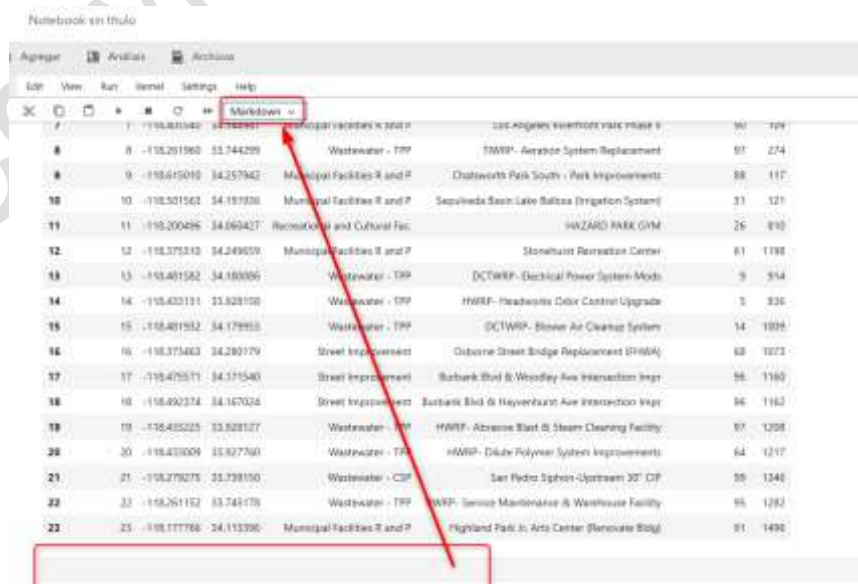
Paso 4: Markdown

Markdown es un lenguaje de marcado ligero que permite la escritura sencilla de texto que luego es convertido a formato HTML y por lo tanto queda bien reproducido en el notebook.

Este lenguaje fue creado por John Gruber y puede consultarse toda su sintaxis en su web: <https://daringfireball.net/projects/markdown/syntax>

ArcGIS Notebooks como Notebooks dentro del entorno ArcGIS Pro soporta notación Markdown en sus celdas.

- Para una nueva celda seleccione en el menú desplegable *Markdown*:



Va a utilizar sintaxis básica de Markdown para crear un contenido más visual en nuestros notebooks.

- En una celda marcada como Markdown escriba:

```
# Este es título del pie de página de nuestro primer Notebook
## Subtítulo?

> Además, podemos añadir ciertos textos comentados en diferentes líneas.

> Podemos generar tantas líneas como se considere.
```

- Ejecute pulsando el botón Run o con atajos de teclado (Shift+Enter/Ctrl+Enter).

▾ Este es título del pie de página de nuestro primer Notebook

Subtítulo?

Además, podemos añadir ciertos textos comentados en diferentes líneas.

Podemos generar tantas líneas como se considere.

Además, puede “mezclar” Markdown con HTML.

- Haga doble click sobre la celda de Markdown para abrir el modo edición y añada un salto de línea con el código **
** junto con la incorporación de una imagen alojada en un servidor.

<https://www.esri.es/content/dam/distributor-share/esri-es/formacion/experto-gis/bn-masterol.jpg>

```
# Este es título del pie de página de nuestro primer Notebook
## Subtítulo?

> Además, podemos añadir ciertos textos comentados en diferentes líneas.

> Podemos generar tantas líneas como se considere.

<br />
<img src = "https://www.esri.es/content/dam/distributor-share/esri-es/formacion/experto-gis/bn-masterol.jpg">
```

- Ejecute la celda (Shift+Enter).

- Este es título del pie de página de nuestro primer Notebook

Subtítulo?

Además, podemos añadir ciertos textos comentados en diferentes líneas.

Podemos generar tantas líneas como se considere.



Se puede representar las tablas en Markdown mediante la siguiente sintaxis:

```
## Representación de las tablas
```

```
|Runtime Version | ArcGIS Enterprise | ArcGIS Online |
```

```
|:---: | :---: | ---: |
```

```
|12.0 | 11.5 | Mayo 2025 |
```

```
|11.0 | 11.4 | Noviembre 2024 |
```

```
|10.0 | 11.3 | Junio 2024 |
```

```
|9.0 | 11.2 | Octubre 2023 |
```

```
## Representación de las tablas
```

```
|Runtime Version | ArcGIS Enterprise | ArcGIS Online |
```

```
|:---: | :---: | ---: |
```

```
|12.0 | 11.5 | Mayo 2025 |
```

```
|11.0 | 11.4 | Noviembre 2024 |
```

```
|10.0 | 11.3 | Junio 2024 |
```

```
|9.0 | 11.2 | Octubre 2023 |
```

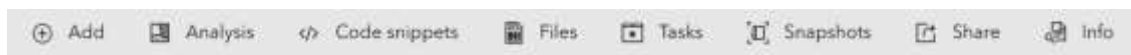
➤ Ejecute la celda de Markdown para poder visualizar la tabla

Representación de las tablas ¶

Runtime Version	ArcGIS Enterprise	ArcGIS Online
12.0	11.5	Mayo 2025
11.0	11.4	Noviembre 2024
10.0	11.3	Junio 2024
9.0	11.2	Octubre 2023

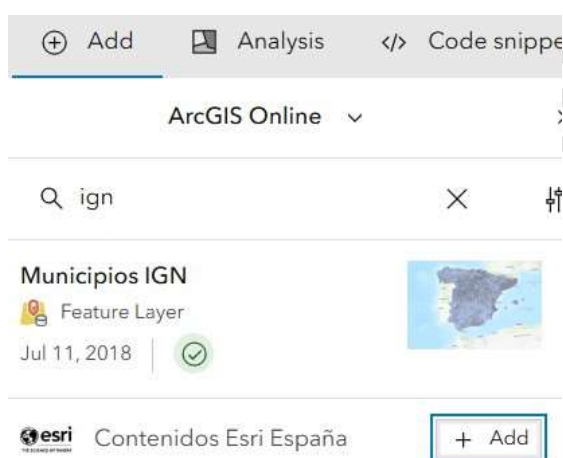
Paso 5: Uso de la cinta de ArcGIS Notebooks en ArcGIS Online

El funcionamiento y la lógica de Jupyter Notebook y los ArcGIS Notebook sea muy parecido, podemos ver una diferencia clara en su apariencia y es que en los ArcGIS Notebook de ArcGIS Online y ArcGIS Enterprise hay una cinta con funcionalidades extra que no tenemos en los clásicos Jupyter Notebook.

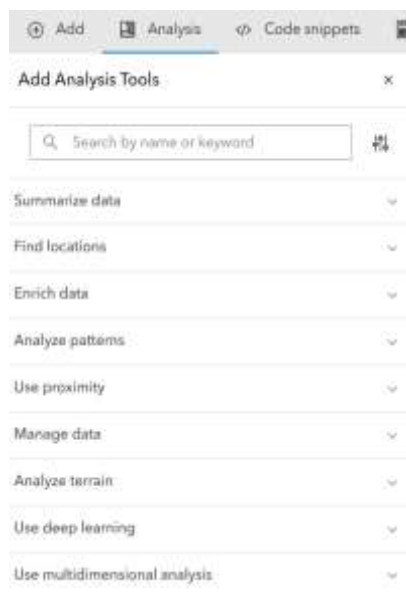


Esta cinta nos facilita la interacción y la creación de código ya que podemos hacer cosas como añadir contenido o analizar datos directamente con una interfaz.

- El botón de **añadir** nos permite buscar y agregar contenido dentro de nuestro contenido, de nuestra organización, del Living Atlas y de ArcGIS Online.



- Desde **análisis** podemos añadir fácilmente el código para agregar diferentes análisis como resumen de puntos o encontrar localizaciones.



- Sin embargo, si hemos tenido que desarrollar código, podemos guardarlo y compartirlo con el resto desde la pestaña de **code snippets**. Desde ella podremos acceder a fragmentos de código creados por Esri, por nosotros mismos o por algún miembro de nuestra organización.
- Desde **archivos** podemos subir archivos de forma temporal (no se subirán como un ítem en el portal) para tener acceso a ellos desde el notebook.
- El botón de **tareas** nos permite programar la ejecución de los notebooks. De esta forma podemos, por ejemplo, hacer que un notebook actualice una capa de datos una vez al día o cada 15 minutos. Los ArcGIS Notebooks tienen distintos periodos de programación y varía entre ArcGIS Online y ArcGIS Enterprise.
- La pestaña de **snapshots** nos permite crear "copias de seguridad" del notebook porque nos permite volver a versiones anteriores. Esto es muy útil si estamos haciendo cambios importantes y no estamos seguros de si funcionarán o no.
- El botón de **compartir** nos permite definir con quién queremos compartir el elemento, sin necesidad de salir de esta interfaz de usuario.

Información del elemento x

Título  Editar

Ejercicio 1A: ArcGIS Notebooks

General

Tipo: Notebook

Propietario: [yulia.streltsova_educacionEsri](#)

Visualizaciones: 2

Creada: 17 dic 2025

Actualizado: 18 dic 2025

Tiempo de ejecución: [ArcGIS Notebook Python 3 Standard - 12.0](#)

Descripción  Editar

[Agregue una descripción detallada del elemento.](#)

Etiquetas  Editar

Este elemento no tiene ninguna etiqueta.

Condiciones de uso  Editar

[Agregue restricciones especiales, exenciones de responsabilidad, términos y condiciones o limitaciones sobre el uso del contenido del elemento.](#)

○ Por último, la pestaña de **información** nos detalla los aspectos más específicos del notebook como su título, descripción y etiquetas, así como el runtime con el que se ha creado el notebook, cuándo se ha modificado, etc.

➤ Guarde el Notebook como **Ejercicio 1: ArcGIS Notebooks**

2. Herramientas Python de utilidad

Python para la ingeniería de datos

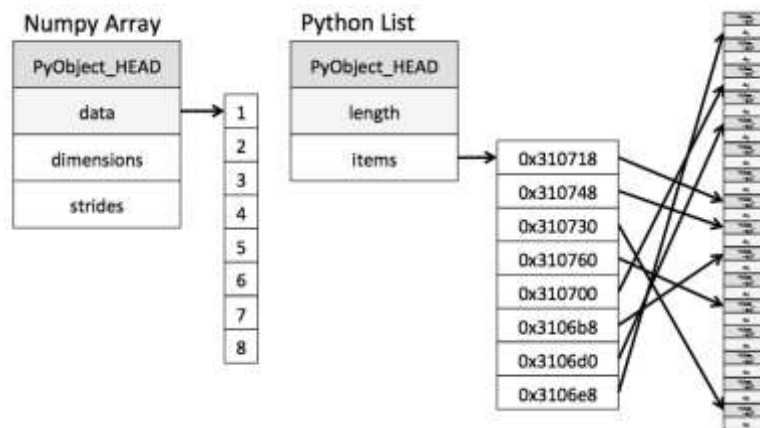
A partir de SciPy, un conjunto de librerías open source orientadas al cálculo científico, se ha desarrollado un amplio ecosistema de herramientas y tecnologías en Python que facilitan el trabajo en ingeniería y análisis de datos. (<https://www.scipy.org/topical-software.html>)



En el contexto del uso del API de Python, resultan especialmente relevantes las librerías centradas en el procesamiento de datos (como NumPy, SciPy o Pandas) y en su visualización (como Matplotlib, Seaborn o Bokeh). A continuación, se revisarán algunas de ellas, dado que constituyen recursos fundamentales a la hora de abordar proyectos GIS con Python.

NumPy

Cuando se trabaja con grandes volúmenes de datos organizados en listas o matrices, Python puede experimentar una pérdida de rendimiento debido a su arquitectura interna y a su tipado dinámico y débil.



Para evitar esta pérdida de rendimiento, surgió NumPy, una librería diseñada para trabajar con arrays n-dimensionales de forma eficiente. Además de optimizar la gestión de grandes volúmenes de datos numéricos, incorpora un conjunto amplio de funciones pensadas específicamente para este tipo de operaciones.

De forma predeterminada, Python gestiona las matrices mediante listas (list). Esto puede generar cierta confusión, ya que también existe el módulo estándar array, que permite trabajar con secuencias tipadas de elementos. Aunque comparten nombre con los arrays de NumPy, su propósito y capacidades difieren notablemente.

Pandas

Pandas es una librería construida sobre NumPy que proporciona estructuras diseñadas para facilitar el análisis de datos. Las dos estructuras principales son:

- **Series:** arrays unidimensionales con índices.
- **DataFrames:** arrays bidimensionales similares a una tabla SQL o una hoja de Excel.

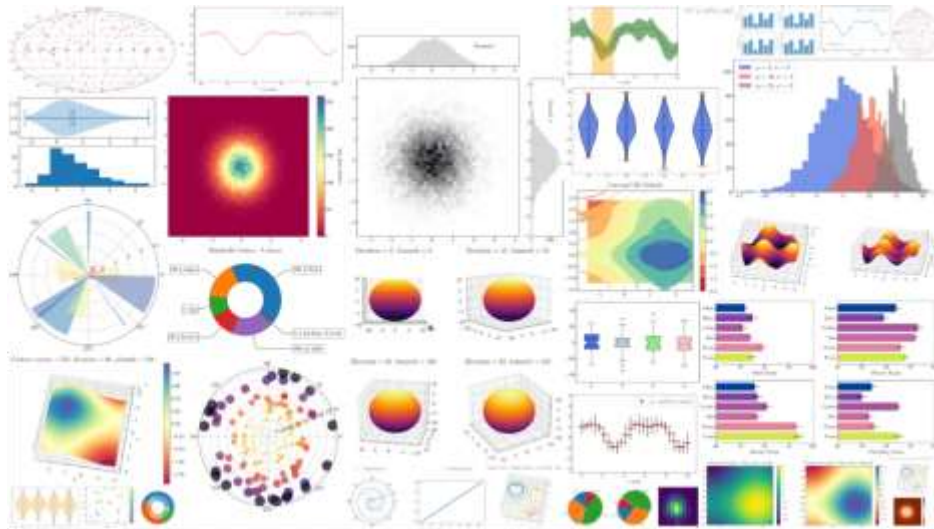
Ambos tipos se apoyan internamente en las matrices de NumPy y constituyen el núcleo del modelo de datos de Pandas. Mientras que NumPy está pensado para manejar datos homogéneos y bien estructurados, Pandas resulta más adecuado para datos tabulares o heterogéneos. NumPy funciona de manera excelente cuando los datos están limpios y ordenados; sin embargo, sus límites aparecen al trabajar con información menos estructurada donde necesitamos más flexibilidad, como asignar etiquetas, gestionar datos faltantes, agrupar, pivotar o transformar.

Pandas combina la eficiencia de la computación matricial de NumPy con la versatilidad de las bases de datos relacionales. Ofrece una indexación potente para remodelar, dividir, agregar o seleccionar subconjuntos de datos, además de herramientas para manejar valores nulos, añadir o eliminar columnas, imputar información y generar gráficos de forma inmediata. En definitiva, Pandas es una pieza fundamental para cualquier flujo de trabajo serio de manipulación y análisis de datos.

Matplotlib

En ciencia de datos es habitual complementar el análisis con visualizaciones gráficas, ya que permiten detectar patrones y tendencias de un vistazo. Dentro del ecosistema SciPy –un conjunto de librerías científicas conocido como *Scientific Python*–

pág. 20



Ejercicio 2: Conociendo las librerías importantes del uso de API de Python

Paso 1: Comenzando con NumPy

- Importe la librería que ya tenemos en ArcGIS Notebooks como np:

```
[1]: import numpy as np
```

Nota: Existe la convención de dar **"np"** como el nombre que se asigna a numpy para su uso. Este tipo de convenciones no es necesario seguirlas, pero es muy común encontrar gran cantidad de scripts que la usan.

- Convierta una lista convencional de Python en un array de NumPy utilizando el siguiente código:

```
[2]: arr = [1,2,3,4,5] #array definitivo normalmente en python(List)
      numArr = np.array(arr)
      numArr
```

```
[2]: array([1, 2, 3, 4, 5])
```

Nota: la función **np.array()** sirve para crear un array a partir de una lista de Python.

- Pruebe obtener el mismo resultado con la función **np.arange**

```
np.arange(1,6)
```

```
array([1, 2, 3, 4, 5])
```

Nota: np.arange(1, 6) genera un **array de NumPy** con una secuencia de valores **desde 1 hasta 5**, incrementando de 1 en 1.

También puede fijar el tipo de elementos que forman el array.

- Repita el paso anterior para generar un array con la función **np.array** pero indicando el tipo de datos como **float32**

```
np.array([1,2,3,4,5], dtype='float32')
```

```
array([1., 2., 3., 4., 5.], dtype=float32)
```

Lo tipos de [datos aceptados por NumPy](#) son:

Data type	Description
<code>bool_</code>	Boolean (True or False) stored as a byte
<code>int_</code>	Default integer type (same as C <code>long</code> ; normally either <code>int64</code> or <code>int32</code>)
<code>intc</code>	Identical to C <code>int</code> (normally <code>int32</code> or <code>int64</code>)
<code>intp</code>	Integer used for indexing (same as C <code>ssize_t</code> ; normally either <code>int32</code> or <code>int64</code>)
<code>int8</code>	Byte (-128 to 127)
<code>int16</code>	Integer (-32768 to 32767)
<code>int32</code>	Integer (-2147483648 to 2147483647)
<code>int64</code>	Integer (-9223372036854775808 to 9223372036854775807)
<code>uint8</code>	Unsigned integer (0 to 255)
<code>uint16</code>	Unsigned integer (0 to 65535)
<code>uint32</code>	Unsigned integer (0 to 4294967295)
<code>uint64</code>	Unsigned integer (0 to 18446744073709551615)
<code>float_</code>	Shorthand for <code>float64</code> .
<code>float16</code>	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
<code>float32</code>	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
<code>float64</code>	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
<code>complex_</code>	Shorthand for <code>complex128</code> .
<code>complex64</code>	Complex number, represented by two 32-bit floats
<code>complex128</code>	Complex number, represented by two 64-bit floats

Es muy habitual crear datos de ejemplo usando el módulo `random` (<https://numpy.org/doc/stable/reference/random/index.html>).

- Cree un array con los números aleatorios utilizando la función **`randint`** y ejecute la celda:

```
#crear datos random
np.random.randint(1,100,8)

array([50, 94, 46, 18, 81, 87, 48, 22])
```

Nota: el resultado de la salida puede ser diferente en cada ordenador dado que esa función crea los números y el orden aleatoriamente.

- Cree una *matriz de números enteros aleatorios* usando **`np.random.randint()`**.

```
np.random.randint(1, 100, (2, 3))
```

Que significa: **1** → valor mínimo **incluido**, **100** → valor máximo **excluido**, **(2, 3)** → tamaño de la matriz (2 filas, 3 columnas). Es decir, genera una matriz 2×3 con enteros aleatorios entre **1 y 99**.

```
#crear matrices random
np.random.randint(1,100,(2,3))

array([[37, 47, 64],
       [53,  7, 22]])
```

Nota: la función `np.random.randint(low, high=None, size=None, dtype=int)`

Devuelve números enteros aleatorios desde low (incluido) hasta high (excluido). Devuelve números enteros aleatorios desde la distribución «uniforme discreta» del tipo de datos especificado en el intervalo «semiabierto» [low, high). Si high es None (el valor predeterminado), los resultados son desde [0, low): [numpy.random.randint – NumPy v2.3 Manual](#)

- Guarde el Notebook para no perder los cambios con el Nombre **“Ejercicio 2: Conociendo las librerías de API_iniciales”**
- Mejore el script agregando el título, subtítulos correspondientes a los pasos y comentarios adicionales:

Este notebook explica el uso de las librerías importantes de API de Python.

- Comenzando trabajar con Numpy
- Uso de la librería pandas
- Comenzar a trabajar con Matplotlib

▼ Conectar con el portal GIS

```
In: from arcgis.gis import GIS
gis = GIS("home")
```

Conocer la librería Numpy

```
In: import numpy as np

In: arr = [1,2,3,4,5] #array definitivo normalmente en python(List)
numArr = np.array(arr)
numArr

In: array([1, 2, 3, 4, 5])

In: np.arange(1,6)

In: array([1, 2, 3, 4, 5])

In: np.array([1,2,3,4,5], dtype='float32')

In: array([1., 2., 3., 4., 5.], dtype=float32)

In: #crear datos random
np.random.randint(1,100,8)
```

Paso 2: Comenzando con Pandas

Al estar basado en NumPy, la integración es directa. Primero, va a ver cómo se construye un array a través de una de las estructuras de datos más conocida de la biblioteca Pandas, Series.

- Inserte nueva celda de Markdown para dar la instrucción sobre el siguiente paso:

Comenzando con Pandas

- Importe las bibliotecas: pandas como pd y numpy como np:

Comenzando con Pandas

```
[1]: import pandas as pd
import numpy as np
```

Explorar el formato Series

- Cree un array de los días de semana y luego conviértelo en el formato Series:

```
[ ]: semana = np.array(['lunes', 'martes', 'miercoles', 'jueves', 'viernes'])
serie = pd.Series(semana)
serie
```

- Ejecute la celda.

```
[2]: semana = np.array(['lunes', 'martes', 'miercoles', 'jueves', 'viernes'])
serie = pd.Series(semana)
serie

[2]: 0    lunes
1    martes
2  miercoles
3     jueves
4     viernes
dtype: object
```

- Acceda a los diferentes elementos según su indexación:

```
[3]: #Acceder a diferentes elementos utilizando los indices
serie[0]
```

```
[3]: 'lunes'
```

Sugerencia: Practica por tu cuenta con diferentes índices para ver como el código está devolviendo el resultado

También es posible modificar este índice para los elementos que componen la serie, en realidad, el par índice-valor forman un array de numpy.

- Haga una nueva variable finde a partir de una lista ['sabado', 'domingo'] con los índices ['a', 'b'] y ejecute la celda:

```
[6]: #Modificar los indices que componen la serie el par indice-valor que forman un array de numpy
finde = pd.Series(['sabado', 'domingo'], index=['a','b'])
finde

[6]: a    sabado
     b    domingo
     dtype: object
```

Cuando trabajamos con pandas dentro de ArcGIS Notebooks (por ejemplo, después de convertir una capa a DataFrame), es muy común encontrarnos con **Series**, que son columnas individuales con un índice.

Una Serie es como una lista, pero cada valor tiene una etiqueta de índice asociada.

Tenemos dos elementos:

- índice 'a' → valor 'sabado'
- índice 'b' → valor 'domingo'

En pandas, existen **tres formas** de acceder a los elementos:

Forma	Qué busca	Ejemplo	Recomendado
serie[pos]	HOY busca la posición, pero está en desuso	serie[-1]	✗ No recomendable
serie.iloc[pos]	Posición numérica	serie.iloc[-1]	✓ Correcto
serie.loc[indice]	Etiqueta del índice	serie.loc['b']	✓ Correcto

- Acceda por posición (último elemento) usando `finde.iloc[-1]`:

```
[8]: finde.iloc[-1]

[8]: 'domingo'
```

También es posible jugar con diferentes series.

- Cree 2 series independientes con el siguiente código:

```
[9]: s1 = pd.Series(['a', 'b', 'c'])
     s2 = pd.Series(['d', 'e', 'f'])
```

- Anexe una serie con la otra con el método `concat` con el siguiente código:

```
s3 = pd.concat([s1,s2])

s3
```

```
[11]: s3 = pd.concat([s1,s2])
      s3
```

```
[11]: 0    a
      1    b
      2    c
      0    d
      1    e
      2    f
      dtype: object
```

Nota: Previamente utilizaron el método `.append()` para anexas series. En las nuevas versiones de APIs se ha sido sustituido por **`pandas.concat()`**. Porque **`series.append()`** era: lento, redundante, y hacía lo mismo que `concat()` pero con peor rendimiento. Ahora pandas quiere una única forma clara de concatenar objetos.

También debe tener en cuenta la posible integridad de los índices, ya que, como se observa en el ejemplo anterior, pueden solaparse y generar dificultades posteriormente. Para reiniciar el índice, puede utilizar el parámetro `ignore_index=True`, lo que resuelve este problema.

- Concatene de nuevo 2 series a través de la variable S3 utilizando el parámetro `ignore_index`:

```
[12]: s3 = pd.concat([s1,s2], ignore_index=True)
      s3
```

```
[12]: 0    a
      1    b
      2    c
      3    d
      4    e
      5    f
      dtype: object
```

Por lo tanto, podríamos decir que una Serie es un array de numpy acompañado con un índice.

Nota: Además, Pandas aporta una batería de atributos y funciones adicionales para tratar con estos elementos: <https://pandas.pydata.org/pandas-docs/stable/reference/index.html>

Por otro lado, es posible crear series de fechas fácilmente.

- Cree una Serie de fechas entre 01/12/2025 y 31/12/2025 y muestre solo los primeros 10 días:

```
[13]: fechas=pd.date_range('01/12/2025', '31/12/2025', freq='D')
      fechas[0:10]
```

```
[13]: DatetimeIndex(['2025-01-12', '2025-01-13', '2025-01-14', '2025-01-15',
                    '2025-01-16', '2025-01-17', '2025-01-18', '2025-01-19',
                    '2025-01-20', '2025-01-21'],
                    dtype='datetime64[ns]', freq='D')
```

Explorar el formato DataFrame

Por otro lado, Pandas proporciona una segunda estructura de datos fundamental: el DataFrame. Se trata de un objeto bidimensional que organiza la información en filas y columnas, de forma muy similar a una tabla de base de datos o a una hoja de cálculo de Excel. Cada columna puede tener un tipo de dato diferente, y cada fila representa un registro. Esta estructura permite realizar operaciones complejas de filtrado, agregación, limpieza o transformación de manera eficiente y expresiva, convirtiendo al DataFrame en la herramienta central de Pandas para el análisis y manipulación de datos tabulares.

Para crear un DataFrame, Pandas permite partir de diversas fuentes de datos. Entre las más habituales se incluyen:

- **Listas de Python**, útiles para construir estructuras sencillas o datos generados manualmente.
- **Series de Pandas**, que pueden combinarse para formar un DataFrame completo.
- **Arrays de NumPy (ndarray)**, especialmente útiles cuando los datos ya están estructurados numéricamente.
- **Otros DataFrames**, lo que facilita concatenar, fusionar o transformar datos provenientes de múltiples tablas.

A continuación, se muestra cómo crear un **DataFrame** a partir de una lista de listas en Python. En este ejemplo, especificará también los nombres de las columnas para estructurar correctamente la información:

```
[14]: #Trabajar con DataFrame
datos = [['Jorge', 26], ['Miguel', 41], ['Diego', 36]]
df = pd.DataFrame(datos, columns=['Nombre', 'Edad'])
df
```

```
[14]:
```

	Nombre	Edad
0	Jorge	26
1	Miguel	41
2	Diego	36

La salida presenta un DataFrame con índices generados automáticamente, comenzando en cero y aumentando de uno en uno.

Del mismo modo, es posible definir manualmente los **índices** del DataFrame. Esto resulta especialmente útil cuando desea utilizar etiquetas descriptivas para identificar cada fila:

```
[15]: df = pd.DataFrame(datos, columns=['Nombre', 'Edad'], index=['profesor 1', 'profesor 2', 'profesor 3'])
df
```

```
[15]:
```

	Nombre	Edad
profesor 1	Jorge	26
profesor 2	Miguel	41
profesor 3	Diego	36

- Cree un DataFrame personalizado cambiando los parámetros como nombre, edad y los nombres de índices según el ejemplo anterior.

Crear un DataFrame a partir de Series de Pandas

Además de listas o arrays de NumPy, un DataFrame puede construirse directamente a partir de **Series de Pandas**. En el ejemplo siguiente se crean dos Series y posteriormente se combinan en un único DataFrame asignando a cada una un nombre de columna:

```
[16]: #Crear un DataFrame a partir de Series de Pandas
s1 = pd.Series(['a', 'b', 'c'])
s2 = pd.Series(['d', 'e', 'f'])
datos = {'columna1': s1, 'columna2': s2}
df = pd.DataFrame(datos)
df
```

```
[16]:
```

	columna1	columna2
0	a	d
1	b	e
2	c	f

- Copie el siguiente código para crear un data Frame. *Opcionalmente, se puede cambiar valores para obtener un Data Frame personalizado.*

```
#Crear un DataFrame a partir de Series de Pandas
s1 = pd.Series(['a', 'b', 'c'])
s2 = pd.Series(['d', 'e', 'f'])
datos = {'columna1': s1, 'columna2': s2}
df = pd.DataFrame(datos)
df
```

En este ejemplo cada Serie se convierte en una columna del DataFrame, y los índices de ambas Series se alinean automáticamente.

Gestión de datos faltantes (NaN)

Cuando las Series tienen longitudes diferentes, Pandas completa la información faltante con el valor especial NaN (Not a Number), que se utiliza para representar datos ausentes o no disponibles:

```
[17]: s1 = pd.Series(['a', 'b', 'c'])
      s2 = pd.Series(['d'])
      datos = {'columna1': s1, 'columna2': s2}
      df = pd.DataFrame(datos)
      df
```

```
[17]:
```

	columna1	columna2
0	a	d
1	b	NaN
2	c	NaN

- Copie el siguiente código para obtener un Data Frame con los datos faltantes. *Opcionalmente, se puede cambiar valores para obtener un Data Frame personalizado:*

```
s1 = pd.Series(['a', 'b', 'c'])
s2 = pd.Series(['d'])
datos = {'columna1': s1, 'columna2': s2}
df = pd.DataFrame(datos)
df
```

Selección de columnas

Aunque trabaje con DataFrames, es posible seleccionar una única columna del mismo modo que accederíamos a un diccionario en Python. La columna seleccionada se devuelve como un objeto Series:

```
[18]: df['columna1']
```

```
[18]:
```

0	a
1	b
2	c

```
Name: columna1, dtype: object
```

- Devuelva los valores correspondientes del DataFrame según los nombres que hayan definido previamente.

Creación de nuevas columnas

Pandas permite añadir nuevas columnas simplemente asignando una Serie o un valor a un nombre de columna. Si la Serie es más corta que el DataFrame, Pandas completará automáticamente el resto de las filas con NaN:

```
[21]: #Creación de nuevas columnas
df['columna3'] = pd.Series(['x', 'y'])
df
```

```
[21]:
```

	columna1	columna2	columna3
0	a	d	x
1	b	NaN	y
2	c	NaN	NaN

- Personalice el ejemplo modificando los nombres de las columnas y los valores de la Serie.

Eliminación de columnas

Cada columna de un DataFrame existe como un objeto independiente en memoria, y puede eliminarse fácilmente utilizando la instrucción del:

```
[22]: del df['columna2']
df
```

```
[22]:
```

	columna1	columna3
0	a	x
1	b	y
2	c	NaN

- Personalice y adapte el ejemplo según los valores elegidos anteriormente.

El uso de la función read_csv()

Una de las funciones más utilizadas en Pandas es **read_csv()**, ya que la transferencia de información mediante ficheros CSV es hoy en día muy común. Poder leer este tipo de archivos de manera ágil y eficiente es fundamental en cualquier flujo de análisis de datos.

Para ilustrarlo, a continuación, se muestra cómo leer un fichero CSV que contiene información de los municipios de Gran Canaria. En este caso, el archivo se encuentra publicado en ArcGIS Online y puede accederse directamente mediante su URL: <https://esrimasteronline.maps.arcgis.com/sharing/rest/content/items/c5eee6b92e1a4a74b892e512db8e04dd/data>

- Cree una variable *nucleos* y con la función **read.csv()** abra el fichero:

```
#Uso de la funcion read_csv()

nucleos = pd.read_csv('https://esrimasteronline.maps.arcgis.com/sharing/rest/content/items/c5eee6b92e1a4a74b892e512db8e04dd/data')

nucleos
```

```
[23]: #Uso de la función read_csv()
nucleos = pd.read_csv('https://esriasteronline.maps.arcgis.com/sharing/rest/content/items/c5ee6b92e1a476d891e512db9e84dc/data')
nucleos
```

	FID	OBJECTID	Cod_INE	Texto	Poblacion	CodMun	Municipio	Cod_Provin	Provincia	X	Y
0	0	1	35002000101	Agüimes	29431.0	35002	Agüimes	35	Las Palmas	-15.446912	27.900542
1	1	2	35003000201	Antigua	10458.0	35003	Antigua	35	Las Palmas	-14.013301	28.418966
2	2	3	35004000301	Arrecife	58156.0	35004	Arrecife	35	Las Palmas	-13.551451	28.960649
3	3	4	35006000202	Arucas	36745.0	35006	Arucas	35	Las Palmas	-15.521512	28.117496
4	4	5	35009000303	Gáldar	24473.0	35009	Gáldar	35	Las Palmas	-15.654111	28.143873
5	5	6	35011000201	Ingenio	29640.0	35011	Ingenio	35	Las Palmas	-15.442261	27.918781
6	6	7	35012000201	Mogán	22638.0	35012	Mogán	35	Las Palmas	-15.726229	27.880172
7	7	8	35014000501	La Oliva	22351.0	35014	La Oliva	35	Las Palmas	-13.927340	28.608322
8	8	9	35015000501	Pájara	20622.0	35015	Pájara	35	Las Palmas	-14.105631	28.349752
9	9	10	35016000401	Las Palmas de Gran Canaria	383308.0	35016	Las Palmas de Gran Canaria	35	Las Palmas	-15.413387	28.099775

Del mismo modo, es posible leer otros archivos CSV disponibles públicamente en ArcGIS Online. Por ejemplo, el conjunto de datos **"Hawaii Emergency Shelters CSV"** puede cargarse directamente utilizando su URL:

(<https://www.arcgis.com/sharing/rest/content/items/3f0f20ae77c0447cb5fa2a15038d0520/data>)

```
[24]: #Abrir el csv alojado en ArcGIS Online:
hawaii = pd.read_csv('https://www.arcgis.com/sharing/rest/content/items/3f0f20ae77c0447cb5fa2a15038d0520/data')
hawaii
```

	OBJECTID	NAME	ADDRESS	CITY	STATE	ZIP	PHONE
0	1	Hilo High School	556 Waiuanue Avenue	Hilo	HI	96720	(808) 974-4021
1	2	Holualoa Elementary School	75-5957 Mamalahoa Highway	Holualoa	HI	96725	(808) 322-4800
2	3	Honauanui Elementary School	83-5360 Mamalahoa Highway	Captain Cook	HI	96704	(808) 328-2727
3	4	Hookena Elementary School	86-4355 Mamalahoa Highway	Captain Cook	HI	96704	(808) 328-2710
4	5	Kau High and Pahala Elementary School	96-3150 Pikake Street	Pahala	HI	96777	(808) 313-4100
5	6	Kaunani Elementary School	1710 Kaunani Drive	Hilo	HI	96720	(808) 974-4190
6	7	Kohala Elementary School	54-3609 Akoni Pule Highway	Kapaau	HI	96755	(808) 889-7100
7	8	Waiakewala Elementary School	2420 Kilauea Ave	Hilo	HI	96720	(808) 981-7200
8	9	Hilo Intermediate School	587 Waiuanue Avenue	Hilo	HI	96720	(808) 974-4955

- Guarde el Notebook para no perder los cambios

Paso 3: Comenzando con Matplotlib

Una vez cargados los datos en un DataFrame, podemos aprovechar las capacidades de **Matplotlib**, una de las librerías más utilizadas para la visualización en el ecosistema científico de Python.

- Agregue nueva celda de Markdown y ponga el título "Visualización de datos con Matplotlib en ArcGIS Notebooks" y ejecútela

```
## Visualización de datos con Matplotlib en ArcGIS Notebooks
```

- Importe la librería correspondiente con la siguiente expresión:

```
import matplotlib.pyplot as plt
```

Visualización de datos con Matplotlib en ArcGIS Notebooks

```
[25]: import matplotlib.pyplot as plt
```

ArcGIS Notebooks, además, permite controlar la forma en que se muestran los gráficos mediante las llamadas palabras mágicas de IPython. Estas configuraciones permiten, por ejemplo, generar gráficos embebidos directamente en el cuaderno con una **instrucción especial de IPython** `%matplotlib inline`:

```
%matplotlib inline
```

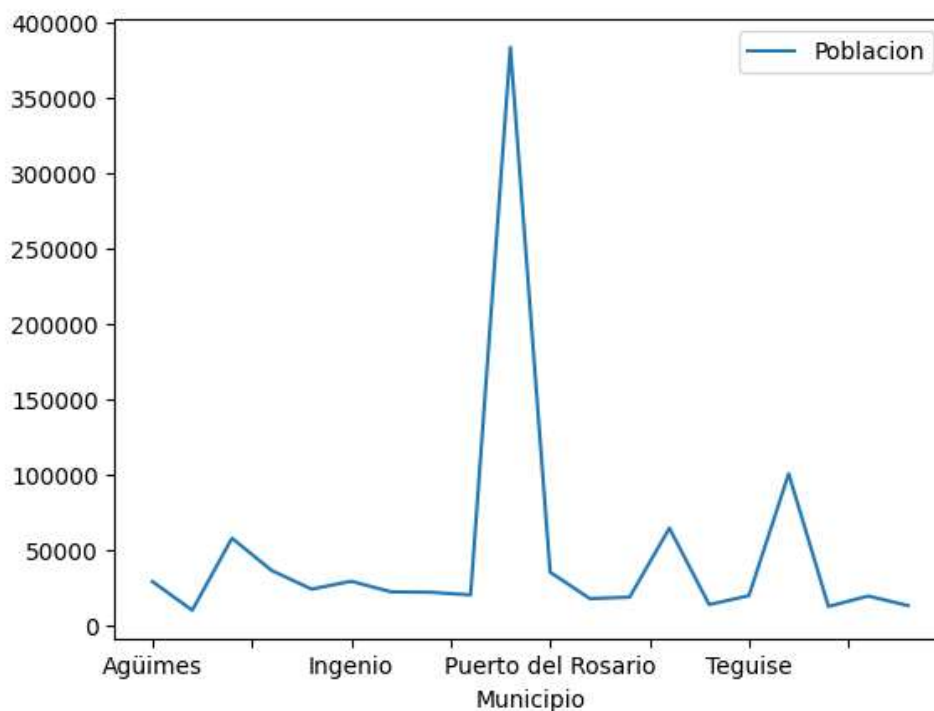
Nota: Para más información: [documentación de IPython](#), [documentación Matplotlib](#), [Visualization with Matplotlib | Python Data Science Handbook](#)

Representación básica de un gráfico

Una vez importada la librería, puede representar datos del DataFrame indicando las columnas que actuarán como eje X y eje Y. En el ejemplo siguiente, se comparan los municipios con su población:

```
[28]: nucleos.plot(x='Municipio', y='Poblacion')
```

```
[28]: <Axes: xlabel='Municipio'>
```



- Inserte la celda de código y escriba la expresión `nucleos.plot(x='Municipio', y='Poblacion')`.
- Ejecute la celda.

Este comando genera, por defecto, una gráfica de líneas. Sin embargo, este tipo de representación no es la más adecuada para comparar valores categóricos (municipios) con valores numéricos (población).

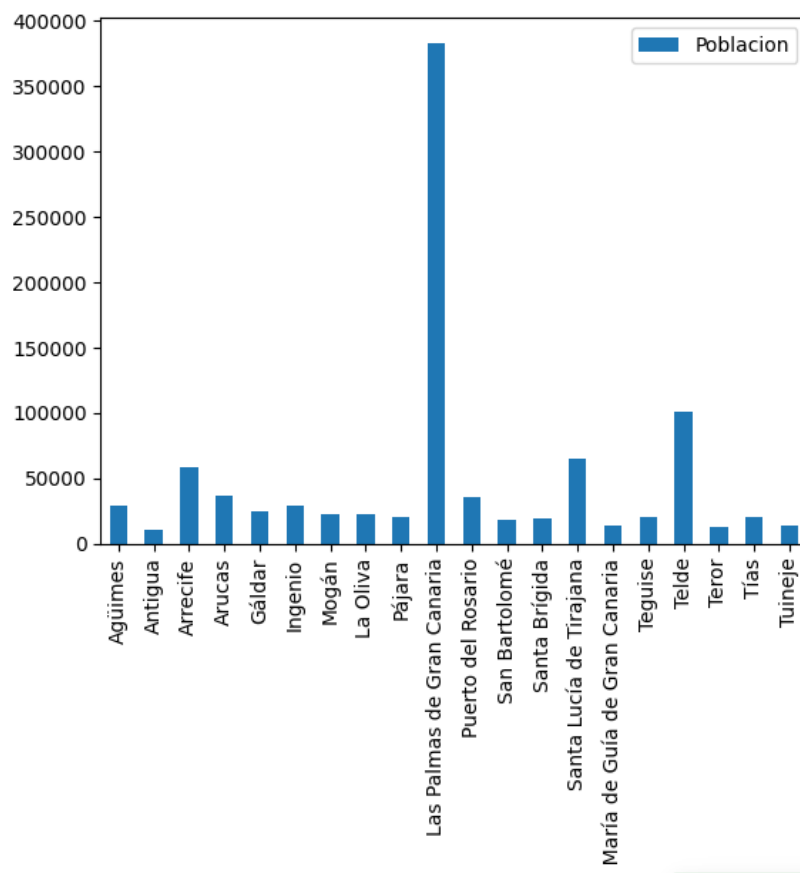
Uso de gráficos de barras

Para representar datos discretos –como poblaciones municipales– es preferible utilizar un gráfico de barras, que permite una lectura mucho más clara.

- En la siguiente celda escriba `nucleos.plot(kind='bar', x='Municipio', y='Poblacion')` y ejecútela:

```
[30]: #grafico de barras
nucleos.plot(kind='bar', x='Municipio', y='Poblacion')
```

```
[30]: <Axes: xlabel='Municipio'>
```

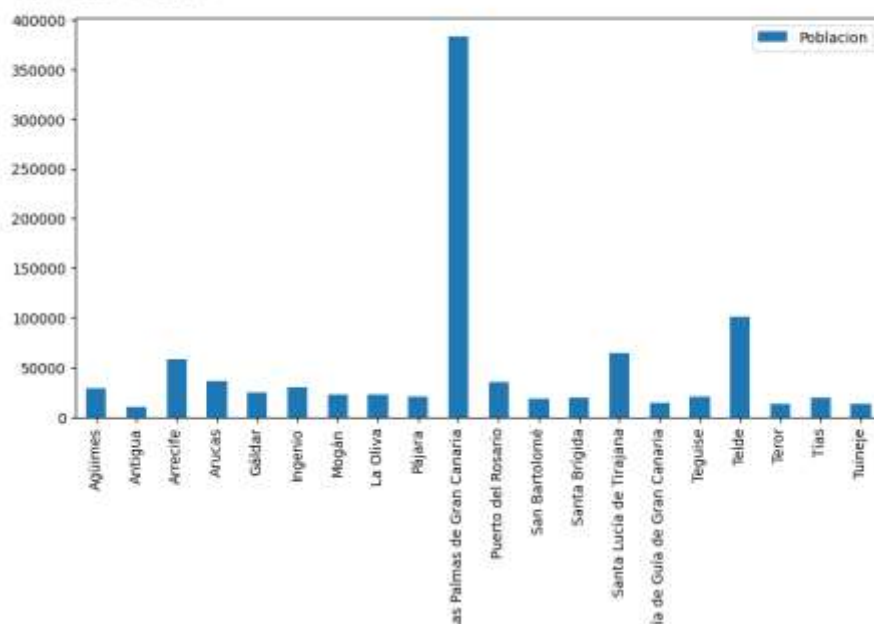


Esta salida ofrece una visualización más adecuada y facilita la comparación entre municipios.

- Ajuste el tamaño del gráfico utilizando el parámetro **figsize=(10,5):**

```
[32]: #Ajustar el tamaño de la figura
nucleos.plot(kind='bar', x='Municipio', y='Poblacion', figsize=(10,5))

[33]: <Axes: xlabel='Municipio'>
```



Se observa que Las Palmas de Gran Canaria presenta un valor muy superior al resto. Esta diferencia acentúa las escalas y dificulta analizar adecuadamente el comportamiento del resto de municipios.

Ajustar el análisis eliminando valores extremos

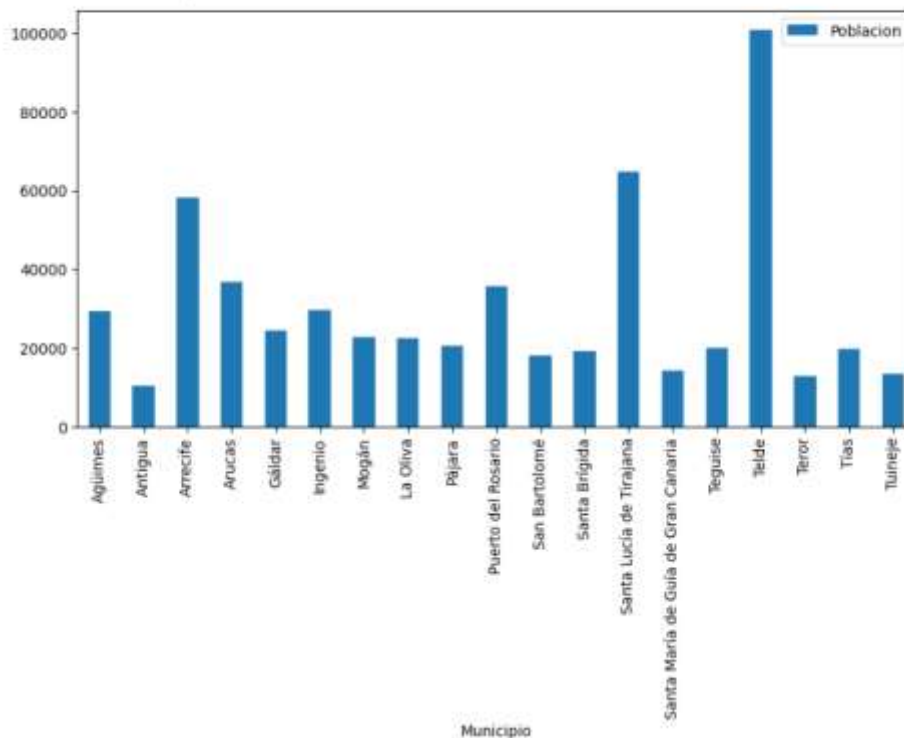
Para obtener una representación más equilibrada, podemos excluir temporalmente este valor extremo y generar un gráfico únicamente con los municipios restantes utilizando la expresión `[ 'Municipio' ] != 'Las Palmas de Gran Canaria' ]`.

➤ Escriba el siguiente código para obtener el resultado óptimo:

```
#Ajustar el analisis eliminando valores extremos
nucleos_sin_lpgc = nucleos[nucleos['Municipio'] != 'Las Palmas de Gran
Canaria']
nucleos_sin_lpgc.plot(kind='bar', x='Municipio', y='Poblacion',
figsize=(10,5))
```

```
[55]: #ajustar el análisis eliminando valores extremos
nucleos_sin_lpgc = nucleos[nucleos['Municipio'] != 'Las Palmas de Gran Canaria']
nucleos_sin_lpgc.plot(kind='bar', x='Municipio', y='Poblacion', figsize=(10,5))

[56]: <Axes: xlabel='Municipio'>
```



Al eliminar el municipio con mayor población, el gráfico muestra de forma homogénea el resto de los valores permitiendo realizar comparaciones más precisas.

La capacidad de generar gráficos directamente desde un DataFrame convierte a Pandas y Matplotlib en herramientas muy potentes dentro de ArcGIS Notebooks. Estas visualizaciones no solo permiten comprender mejor los datos, sino también presentar información de manera clara y accesible para usuarios finales y responsables de toma de decisiones.

- Guarde el Notebook.

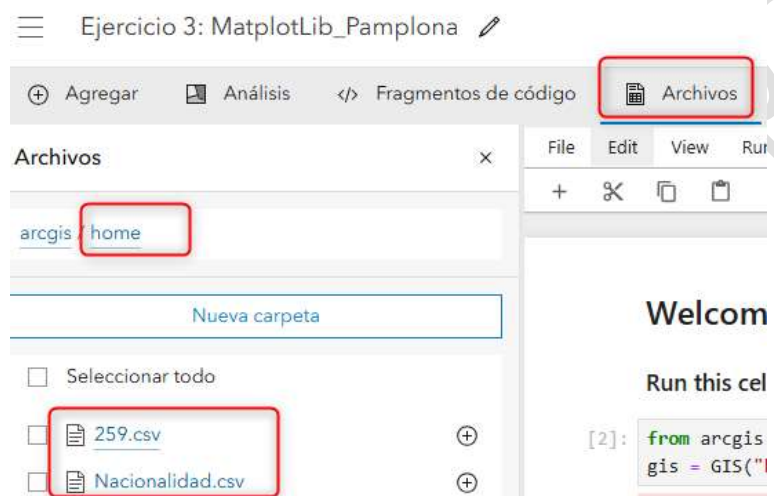
Ejercicio 3: Matplotlib (Pamplona)

El objetivo de este ejercicio es representar en un histograma la población extranjera en el año 2010 en la ciudad de Pamplona.

Paso 1: Representación básica con plot

- Crea un nuevo Notebook llamándolo **"Ejercicio 3: Matplotlib_Pamplona"**
- Cargue los datos en la pestaña de Archivos, primero 259.csv, luego Nacionalidad.csv.

Nota: Los datos se encuentran sobre la siguiente ruta: **C://EsriTraining/PAPI/Files**



- Importe el módulo pandas
- Ejecute la celda una vez cargados los datos del csv 259.csv

```
[1]: import pandas as pd

[ ]: #Ejemplo con error de codificación
dataset = '/arcgis/home/259.csv'
df = pd.read_csv(dataset)
```

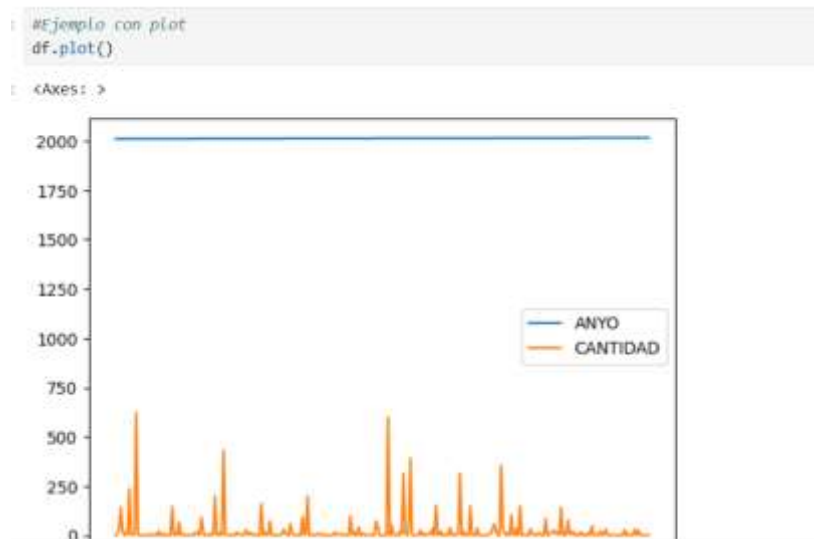
Va a salir el error de codificación. Para solucionarlo, mejoramos el csv, guardando en el formato UTF-8.

Nota: Para esta práctica el csv codificado correctamente ya está preparado como el fichero Nacionalidad.csv

- Ejecute la celda con el fichero correcto con la codificación mejorada

```
[6]: #CSV está corregido con UTF-8
dataset = '/arctis/home/Nacionalidad.csv'
df = pd.read_csv(dataset)
df.head()
```

- Como primer paso, podemos comenzar plotando directamente nuestro DataFrame y ver el resultado con la siguiente sintaxis: `df.plot()`



Paso 2: Eliminar campo innecesario

En el primer paso hemos mostrado todos los registros desde 2010 a 2015 (312 registros). Es el momento de quedarnos solo con los registros de 2010. Podemos obtener un subconjunto de registros directamente del mismo y, una vez obtenidos, podemos eliminar el campo "ANYO" que ya no aporta información para nuestro objetivo:

```
datos2010 = datos[datos.ANYO == 2010]
del datos2010['ANYO']
datos2010.head()
```

	NACIONALIDAD	CANTIDAD
0	ALEMANIA	1
1	ARGELIA	11
2	ARGENTINA	29
3	BOLIVIA	144
4	BRASIL	23

Paso 3: Aplicar el método `set_xticklabels()`

Ahora que ya tenemos un dataset nuevo, podemos plotarlo de nuevo y veremos que, aunque la cantidad (eje y) queda bien referenciado, no tenemos una representación del nombre del país de origen en el eje x.

Debemos cambiar, por tanto, todo lo referente al eje x, tanto los valores como los "ticks", que actualmente muestran valores numéricos de 5 en 5 cuando debería aparecer un "tick" por cada país.

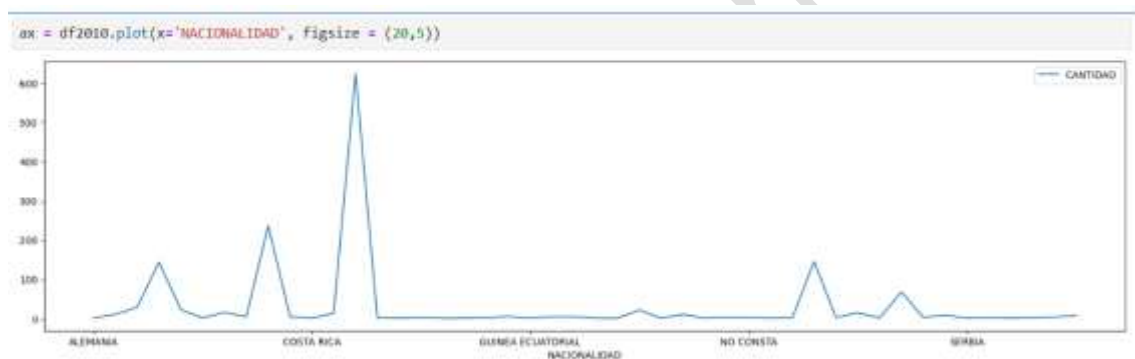
Para ello, vamos a utilizar dos métodos del elemento plotado (ax):

- `set_xticks()`: como parámetro esta función espera como parámetro una lista de los elementos que necesitamos. Los podemos obtener de la propiedad `index` del `DataFrame` (https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.set_xticks.html)
- `set_xticklabels()`: la función anterior hacía las divisiones del eje x mientras que esta función establece los textos que hay debajo de cada una de las "rayitas". En este caso, lo que queremos es mostrar los valores del campo "NACIONALIDAD" de nuestro dataset, es decir: "df2010.NACIONALIDAD"

Nota: Leer más información sobre este método: https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.set_xticklabels.html

➤ Utilice la siguiente sintaxis para llevarlo a cabo:

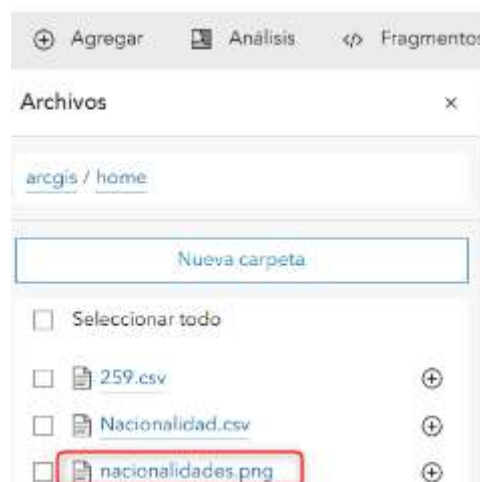
```
ax = df2010.plot(x='NACIONALIDAD', figsize = (20,5))
```



➤ Para guardar el gráfico utilice el método **savefig**:

```
ax.figure.savefig('/arcgis/home/nacionalidades.png')
```

Ahora se puede ver en los archivos dentro de ArcGIS Notebook y guardar posteriormente en el local:



Paso 4: Filtrar registros por año

- Para filtrar registros por un año (por ejemplo, de 2010) en concreto podemos utilizar esta sintaxis:

```
nacionalidades2010 = nacionalidades[nacionalidades.ANYO == 2010]
nacionalidades2010
```

```
nacionalidades2010 = nacionalidades[nacionalidades.ANYO == 2010]
nacionalidades2010
```

	ANYO	NACIONALIDAD	CANTIDAD
0	2010	ALEMANIA	1
1	2010	ARGELIA	11
2	2010	ARGENTINA	29
3	2010	BOLIVIA	144
4	2010	BRASIL	23
5	2010	BURKINA FASO	2
6	2010	CHILE	16
7	2010	CHINA	5
8	2010	COLOMBIA	237
9	2010	CONGO	5
10	2010	COSTA RICA	1

- Para poder visualizar los datos con gráfica de barras elimine la columna ANYO con la función **del**:

```
del nacionalidades2010['ANYO']
nacionalidades2010
```

```
del nacionalidades2010['ANYO']
```

```
nacionalidades2010
```

	NACIONALIDAD	CANTIDAD
0	ALEMANIA	1
1	ARGELIA	11
2	ARGENTINA	29
3	BOLIVIA	144
4	BRASIL	23
5	BURKINA FASO	2
6	CHILE	16
7	CHINA	5
8	COLOMBIA	237
9	CONGO	5
10	COSTA RICA	1
11	CUBA	14

Aunque la gráfica es comprensible, idealmente no debería tener un aspecto lineal ya que se trata de una visualización categórica y, por tanto, deberíamos utilizar un aspecto de gráfica de barras.

- Importe el módulo matplotlib

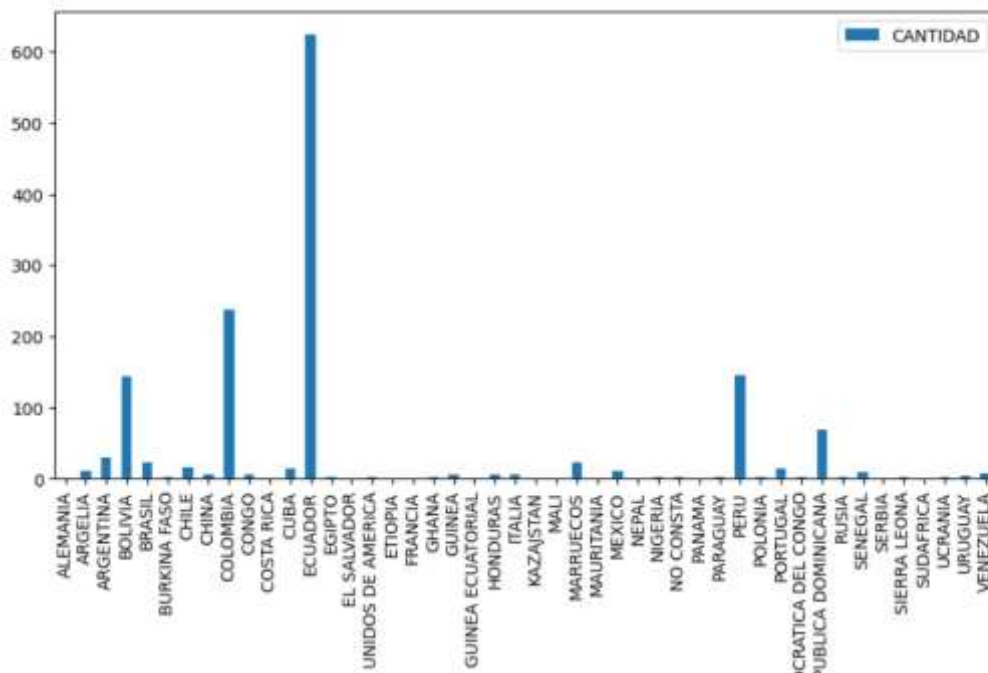
```
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

- Utilice la siguiente sintaxis para poder visualizar como gráfica de barras:

```
ax = nacionalidades2010.plot(kind='bar', x='NACIONALIDAD',  
y='CANTIDAD', figsize=(10,5))
```

```
ax = nacionalidades2010.plot(kind='bar', x='NACIONALIDAD', y='CANTIDAD', figsize=(10,5))
```



- Guarda su gráfica en la ruta de archivos con esta sintaxis:
`ax.figure.savefig('/arctgis/home/nacionalidades2.png')`

```
ax.figure.savefig('/arctgis/home/nacionalidades2.png')
```

3. Introducción a la API de Python

La API de Python es un paquete diseñado para la gestión, administración y uso de mapas y datos espaciales basados en su Web GIS.

Esta API le ofrece herramientas para el análisis de datos vectoriales y ráster, así como para la geocodificación, la creación de mapas o el cálculo de rutas. Además, en el ámbito administrativo, le permite automatizar la creación de usuarios, roles y grupos, así como la generación de ítems (mapas, capas, ficheros, etc.) dentro de su organización Web GIS, ya sea ArcGIS Online o ArcGIS Enterprise.

La API ha sido desarrollado siguiendo las buenas prácticas establecidas por Python, lo que facilita su integración con otras librerías habituales del ecosistema científico, como SciPy, NumPy, Pandas o requests.

Arquitectura

A continuación, realizaremos un repaso de lo que denominamos la arquitectura de la API. En esencia, se trata de la estructura y organización de los módulos que componen el paquete de la API.

Toda esta arquitectura se encuentra descrita en la documentación oficial, dentro de la API Reference: [API Reference for the ArcGIS API for Python | ArcGIS API for Python](#)



La imagen muestra la arquitectura general del ArcGIS API for Python, organizada por bloques funcionales diferenciados mediante colores. Esta representación permite identificar rápidamente las áreas principales de la API y las relaciones entre sus módulos.

Módulo central (verde): GIS

El módulo GIS, representado en color verde, continúa siendo el punto de entrada a su Web GIS. Desde aquí, usted establece la conexión con su organización de ArcGIS

Online o ArcGIS Enterprise y accede a clases esenciales como Users, Groups, Items, Roles, entre otras.

Dentro de este bloque se incluyen **submódulos** relacionados, como **auth** (autenticación) y **env** (entorno de ejecución), que le permiten gestionar la configuración global de la API, tal y como ocurre con los entornos en ArcPy.

Módulos **funcionales (azul claro)**: geoprocesamiento y análisis

Los módulos en tonos azules representan las capacidades de procesamiento, análisis y manipulación de datos geoespaciales. Incluyen:

- Geoprocessing: acceso a herramientas clásicas de análisis.
- GeoAnalytics: análisis distribuido para grandes volúmenes de datos.
- Geoenrichment: enriquecimiento de datos con información demográfica y contextual.
- Geocoding: geocodificación directa e inversa.
- Geometry: operaciones geométricas.
- Network: análisis de redes.
- Graph: estructuras y algoritmos basados en grafos.
- Schematics y Mapping completan este grupo, proporcionando capacidades de visualización estructural y cartográfica.

Estos módulos, en conjunto, representan el núcleo operativo de la API y proporcionan las herramientas necesarias para analizar y transformar la información geográfica.

Nota: Dado que el módulo `arcgis.geoanalytics` ha sido retirado de la API a partir de la versión 2.4.0, es recomendable que no dependan de él en desarrollos nuevos. Si aún utilizan versiones 2.3 o anteriores, asegúrese de que el entorno de su organización lo soporte. Asimismo, explore la migración a GeoAnalytics Engine o utilice los módulos de análisis estándar de la API de Python.

Módulos de datos espaciales (morado): Features, Raster, Realtime

Los módulos en color morado se relacionan con el manejo directo de datos espaciales:

- Features, que incluye clases clave para gestionar entidades vectoriales y sus submódulos asociados.
- Raster, destinado a trabajar con datos ráster y análisis multidimensional.
- Realtime, para flujos de datos en tiempo real.
- En este bloque también aparece SeDF, la estructura Spatially Enabled DataFrame, fundamental para integrar datos espaciales con Pandas.

Módulos de visualización y aplicaciones (naranja): Apps y Widgets

En color naranja se sitúan los módulos orientados a aplicaciones y visualización interactiva:

- Apps: creación y gestión de aplicaciones web.

- Widgets: componentes interactivos especialmente útiles en ArcGIS Notebooks.

Estos módulos permiten construir experiencias visuales y dinámicas para los usuarios finales.

Módulo de aprendizaje (gris): Learn

El módulo Learn, en color gris, recopila utilidades, ejemplos y recursos destinados a facilitar el aprendizaje y la experimentación con La API. Es ideal para quienes desean explorar flujos de trabajo ya preparados.

Módulo independiente (rojo): Notebook

Finalmente, el módulo Notebook, representado en rojo, proporciona clases relacionadas con la gestión y automatización de ArcGIS Notebooks, incluyendo tareas programadas, entornos y ejecución controlada.

Módulos principales

Estos son algunos módulos que resultan interesantes en algunos proyectos:

- **arcgis.gis**

Proporciona funciones y métodos para trabajar con el modelo de geoinformación para los GIS en ArcGIS Online o ArcGIS Enterprise. Este módulo permite gestionar (crear, leer, actualizar y eliminar) usuarios, grupos y contenido.

- **arcgis.env**

Proporciona un entorno compartido utilizado por los diferentes módulos que almacena datos globales, como el SIG activo, parámetros de entorno comunes a todas las herramientas de geoprocésamiento.

- **arcgis.raster**

Una imagen ráster representa una malla de celdas en las que cada una contiene un valor. Normalmente, estos valores son continuos, como en una ortofoto o un MDT. Este módulo contiene clases y funciones que permiten el análisis de este tipo de datos.

- **arcgis.network**

Es muy habitual hoy en día el uso de redes para el análisis no solo de cálculo de rutas, sino de encontrar los más cercanos, gestionar flotas de vehículos, calcular matrices de costes o generar áreas de servicio.

- **arcgis.schematics**

Schematics es un módulo que trata sobre la representación de redes de una forma simplificada y más fácil entendible. Este módulo permite el acceso a este tipo de datos en forma de esquemas.

- **arcgis.geocoding**

La geocodificación, tanto directa como inversa, es una operación muy común en el día a día. El resultado de esta operación puede ser visualizado en un mapa, insertado

como paradas en un cálculo de rutas, usado para crear rutas o como parámetros de entrada para otros análisis. Este módulo provee de las herramientas de acceso a esta funcionalidad de geocodificación así como permite la preparación de scripts para batch geocoding.

- **arccgis.geometry**

Llamamos elemento gráfico a aquel que es visualizado en un mapa. Para ello es necesario que cuente al menos con una geometría y una simbología asociada a la misma. Este módulo define los tipos de geometría que podemos usar y nos provee de funciones para tratar con ellas o convertirlas entre diferentes representaciones.

Nota: Estos son solo algunos de los módulos que forman la API, puedes ver el resto en la documentación oficial: [Overview of the ArcGIS API for Python | ArcGIS API for Python | Esri Developer](#)

Spatially enabled DataFrame

Un **DataFrame** representa, como ya se ha expuesto, una estructura de datos tabular compuesta por una lista de columnas o atributos y una serie de filas o registros, del mismo modo que entendemos una hoja de cálculo o una tabla en una base de datos.

Aprovechando toda la potencia de esta estructura de Pandas, La API de Python incorpora capacidades espaciales adicionales, dando lugar al **Spatially Enabled DataFrame (SeDF)**. Este objeto combina la flexibilidad de Pandas con la capacidad de almacenar, visualizar y analizar información geográfica.

El principal beneficio de los SeDF es que permiten leer y trabajar con recursos espaciales que Pandas, por sí solo, no es capaz de manejar, tales como:

- Shapefiles
- Feature Classes
- GeoJSON
- Feature Layers
- Entre otros formatos espaciales habituales en ArcGIS

Nota: En secciones posteriores se profundizará en este elemento de la API y en las operaciones espaciales que permite realizar: [Introduction to the Spatially Enabled DataFrame | ArcGIS API for Python | Esri Developer](#)

El widget MapView

ArcGIS Notebooks permite la creación de **widgets**, que no son más que representaciones interactivas de distintos objetos dentro del navegador. Por ejemplo, Esri ha desarrollado un widget específico para Notebooks llamado **MapView**, que ofrece al usuario la posibilidad de interactuar directamente con un mapa dentro del propio cuaderno.

Otros ejemplos habituales de widgets son las gráficas interactivas, los visores de tablas con funcionalidades avanzadas o diferentes controles que permiten modificar parámetros en tiempo real. Como podrá imaginar, los widgets son elementos

dinámicos que pueden crearse y configurarse mediante distintos parámetros, proporcionando al usuario final una interfaz mucho más amigable e intuitiva.

Uno de los principales beneficios de ArcGIS Notebooks es que no solo permite ejecutar un script de Python, sino que ofrece al usuario una herramienta interactiva que facilita la comprensión del flujo de trabajo y le permite experimentar con el proceso modificando valores directamente en la interfaz.

La creación de un mapa interactivo dentro de ArcGIS Notebooks es especialmente sencilla gracias a la API de Python. Para ello, basta con generar un objeto de tipo MapView, tal como se muestra en el siguiente ejemplo:

```
from arcgis.gis import GIS
gis = GIS("home")
mapa = gis.map('Spain')
mapa
```

```
[2]: from arcgis.gis import GIS
```

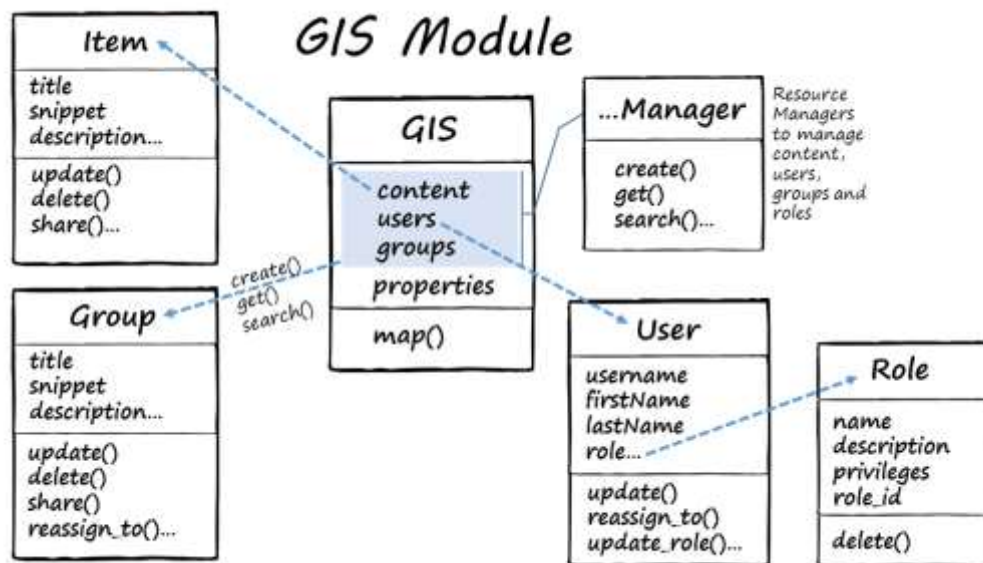
```
[4]: gis = GIS("home")
mapa = gis.map("Spain")
mapa
```



Nota: Relación entre gis y map(): gis es una variable que contiene una instancia de la clase GIS. Es decir, es el objeto que representa su conexión activa al Web GIS (ArcGIS Online o ArcGIS Enterprise). map() es un método, es decir, una función asociada al objeto gis, que permite crear un mapa interactivo dentro de ArcGIS Notebooks. map() solo funciona correctamente cuando se usa sobre la instancia (gis), porque necesita acceder a la sesión, autenticación y propiedades del portal almacenadas en ese objeto.

Módulo GIS

A continuación, se presenta un diagrama del [módulo GIS](#) que muestra las clases más relevantes que lo componen.



Un objeto GIS es la representación de la API de Python de su organización de ArcGIS Online o ArcGIS Enterprise.

Nota: El acceso a su Web GIS, del mismo modo que si accediera desde su navegador, puede realizarse con un usuario de cualquiera de los roles que estén disponibles en su organización. ([User types, roles, and privileges–ArcGIS Online Help | Documentation](#)).

Los usuarios de nivel 1, denominados “Visualizador”, y los usuarios de nivel 2, con capacidades ampliadas –como los roles “Usuario”, “Publicador” y “Administrador”– mantienen en la API las mismas diferencias funcionales que cuando acceden a la plataforma a través de un navegador.

Las restricciones y privilegios que posee cada rol se aplican exactamente del mismo modo al utilizar la API de Python.

Por ello, es posible comprobar estas diferencias creando un objeto GIS autenticado con un usuario de nivel 2 y rol “Usuario”. En este caso, podrá observar que, entre otras limitaciones, dicho usuario no tiene permisos para crear nuevos usuarios, ya que esta acción requiere el rol “Administrador”.

```
[1]: from arcgis.gis import GIS
    gis = GIS("home")

[2]: miGIS = GIS("https://aulaesri.maps.arcgis.com/", "alumnoesri10", "contraseña")

[3]: miGIS.users.create("JuanManuel", "nuevaContra", "nuevo", "usuario", "formacion@esri.es")

Exception: You do not have permissions to access this resource or perform this operation.
(Error Code: 403)
```

Sin embargo, con un rol de “Usuario” sí es posible crear grupos:

```
[4]: miGIS.groups.create("Mi Grupo", "grupoprueba; tagsSiempre")
```

```
[4]:
```

	Mi Grupo Summary: None Description: None Owner: alumnoesri10 Created: November 24, 2025
---	--

El perfil de usuario con el que usted se conecte a su GIS determinará las operaciones que podrá ejecutar mediante código. En una interfaz web, las opciones y botones disponibles se ajustan previamente a sus privilegios. En cambio, en una interfaz de programación, dichos privilegios no se validan hasta que se ejecuta el código, momento en el que el sistema comprueba si su usuario dispone o no de permisos para realizar la operación solicitada.

Patrones en el módulo GIS y en la API

Existen diversas propiedades dentro del módulo GIS que permiten acceder a distintos elementos de su organización, tal y como ya se ha mostrado anteriormente.

Por ejemplo, puede consultar **los usuarios, los ítems y los grupos** de su organización, puesto que estos tres componentes se encuentran expuestos como *propiedades del objeto GIS*. En esencia, se sigue un patrón de tipo proxy, en el que el objeto GIS actúa como intermediario para acceder al resto de elementos del Web GIS.

El acceso a los recursos principales de su Web GIS se realiza siempre a través de estas propiedades, las cuales se implementan sobre una clase abstracta común. Gracias a ello, todos estos elementos comparten la misma estructura de métodos.

Por ejemplo, tanto para buscar usuarios, como grupos o ítems, se utiliza de forma consistente **el método search()**, lo que proporciona una experiencia uniforme y coherente a la hora de interactuar con los distintos componentes del sistema.

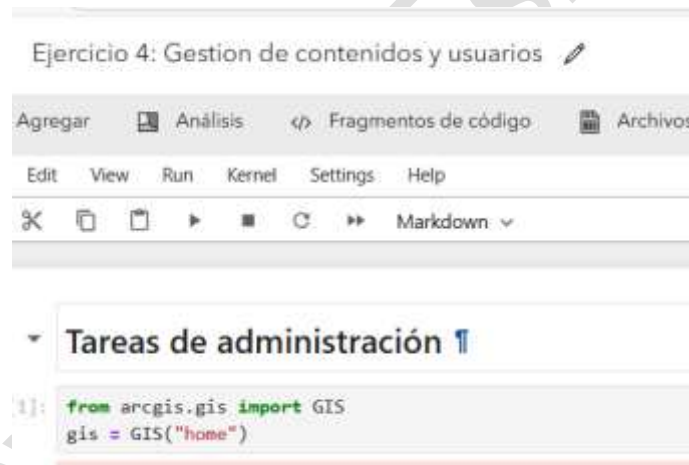
Ejercicio 4: Gestión de contenidos y usuarios

Paso 1: Buscar elementos

Como ya se ha señalado, cuando usted crea un objeto GIS, lo hace empleando un usuario que posee un rol específico, y es precisamente dicho rol el que determina las operaciones que podrá realizar. Además, los resultados obtenidos al ejecutar ciertas operaciones también pueden variar en función de los privilegios asociados a su cuenta.

Por ejemplo, muchos de los elementos creados dentro de una organización –mapas, servicios, capas, archivos, entre otros– son privados por defecto. Debido a ello, un usuario con rol “Usuario” únicamente podrá visualizar sus propios ítems y aquellos que hayan sido compartidos de forma pública. En cambio, un usuario con rol “Administrador” tendrá acceso a todos los elementos de la organización, independientemente de su nivel de privacidad.

- Cree un Nuevo notebook para poder realizar este ejercicio y nómbrelo como **“Ejercicio 4: Gestión de contenidos y usuarios”**:
- Ejecute la primera celda por defecto:



- Para crear un nuevo usuario ejecute la siguiente sintaxis (sustituyendo el nombre, contraseña y correo):

```
gis.users.create("NuevoUsuarioAPI", "NuevaContra", "nuevo", "usuario", "yulia.streltsova@esri.es")
```

```
gis.users.create("NuevoUsuarioAPI", "NuevaContra", "nuevo", "usuario", "yulia.streltsova@esri.es")
Unable to create NuevoUsuarioAPI
Unable to create NuevoUsuarioAPI
```

- Para crear un grupo utilice el método .create cambiando el nombre, la descripción y las etiquetas personalizadas:

```
gis.groups.create("ApiNotebook", "GrupoNuevo; tagsAPIS")
```

```
gis.groups.create("ApiNotebook", "GrupoNuevo; tagsAPIS")
```



ApiNotebook

Summary: None

Description: None

Owner: yulia.streltsova_educacionEsri

Created: December 18, 2025

- Cree una variable `items` para poder acceder al contenido del portal con instancia `content` y el método `search()` y devolviendo la cantidad de ítems con la función `len()`:

```
#Gestionar contenido
item = gis.content.search("*", max_items=10000)
len(item)
```

3976

Nota: La cantidad de ítems obtenida en la búsqueda puede variar en función del rol con el que usted haya iniciado sesión. En este ejemplo, la operación se ha ejecutado empleando un usuario con rol de Administrador, lo que implica que tiene acceso a todos los ítems del portal, incluidos los privados y los pertenecientes a otros usuarios. Si realizara la misma búsqueda con un usuario de rol "Usuario" o "Publicador", el número de ítems devueltos sería previsiblemente menor, ya que únicamente se mostrarían aquellos que sean públicos o que hayan sido compartidos explícitamente con dicho usuario.

*Recuerde que puedes utilizar "MAYUS + TAB" para obtener ayuda sobre el método en ArcGIS Notebooks:

Busqueda de contenidos

```
[2]: items = gis.content.search("*", max_items=10000)
len(items)
```

[2]: 162

[]:

Signature:
gis.content.search(
 query: 'str',
 item_type: 'Optional[str]' = None,
 sort_field: 'str' = 'avgRating',
 sort_order: 'str' = 'desc',
 max_items: 'int' = 10,
 outside_org: 'bool' = False,
 categories: 'Optional[Union[list[str], str]]' =
 category_filters: 'Optional[Union[list[str], str]]' =
 enrich: 'Optional[bool]' = None,
 ...)

Habrá observado que el primer parámetro del método `search()` es siempre `query`. Esto significa que usted puede realizar filtros avanzados que le permitan consultar la información de manera más precisa.

Como valor para este parámetro `query`, es posible utilizar un comodín ("`*`") para recuperar todos los elementos disponibles según los privilegios del usuario con el que haya iniciado sesión.

- Realice la búsqueda de los grupos disponibles al nivel del usuario actual:

```
[3]: #Búsqueda de grupos disponibles dentro del portal
grupos = gis.groups.search("*")
grupos

<Group title:"crimina" owner:marta.davila.edu>,
<Group title:"Crimina2023" owner:educacionEsri>,
<Group title:"Cuadros de Mando Cooperativas" owner:juan.realpe_educacionesri>,
<Group title:"Curso Experience Builder Universidad de Sevilla" owner:ian.gra>,
<Group title:"Deep Learning" owner:cristofer.ramos_educacion>,
<Group title:"demoExperto" owner:educacionEsri>,
<Group title:"Dirección" owner:educacionEsri>,
<Group title:"dn_accenture" owner:educacionEsri>,
<Group title:"ed" owner:educacionEsri>,
<Group title:"Editores Carreteras Diputación León" owner:cristofer.ramos.edu>,
<Group title:"ejemplohub" owner:cristofer.ramos_educacion>,
<Group title:"Ejercicios T1" owner:daniel.fernandez_experto>,
<Group title:"ejie" owner:educacionEsri>,
<Group title:"enagas" owner:educacionEsri>,
<Group title:"Equipo central de Capital Project Tracking" owner:educacionEsri>,
<Group title:"Equipo central de Ciudad Inteligente" owner:educacionEsri>,
<Group title:"Equipo central de clientes REALE" owner:cristofer.ramos_educac:
```

Una vez que usted obtiene la lista de grupos mediante una búsqueda, es posible inspeccionar la clase de cada uno de los objetos devueltos. Cuando ejecuta `grupos = gis.groups.search("*")`, el resultado es: `grupos` → una lista de objetos

Por lo tanto, si desea consultar el nombre de un grupo concreto según su posición dentro de la lista, debe proceder del mismo modo que cuando accede a cualquier elemento de una lista en Python. Es decir, puede utilizar el índice correspondiente, por ejemplo:

```
grupos[12]
```

- Realice varias búsquedas según la posición para ver como comporta el código:

```
<Group title:"Workforce_Asignación_Incidencias_MJPF" owner
```

```
[4]: grupos[12]
```

```
[4]:
```



APP de CAMPOS_OAP

Summary: Grupo para Ineco

Description: None

Owner: esri2240universidades

Created: October 23, 2025

- Guarde el Notebook antes de seguir.

Nota: En ocasiones, el kernel de Python dentro del entorno de **ArcGIS Notebook** puede desconectarse de manera inesperada, lo que implica el riesgo de perder parte del trabajo realizado. Por este motivo, es recomendable **guardar el cuaderno con frecuencia** y, en la medida de lo posible, ejecutar y guardar cada celda tras realizar cambios significativos, evitando así tener que repetir pasos en caso de reinicio del entorno.

Paso 2: Resumen de la actividad de un usuario

En esta actividad trabajará con su propio usuario y generará un registro (log) de su actividad dentro de la organización. El primer paso consiste en localizar su cuenta dentro del portal. Para ello, puede realizar una búsqueda utilizando la propiedad `users` del objeto GIS.

- Acceda a la propiedad `users` con el método `search()`:

```
yo = gis.users.search("nombre_del_usuario")[0]
```

- Ejecute la celda:



Nota: Sustituir el nombre de usuario de cada uno. El método `search()` **devuelve una lista**. El índice `[0]` permite acceder al **primer resultado**, que es el usuario que queremos manipular. Sin `[0]`, estaría trabajando con una lista, no con el objeto `User`.

Sin embargo, existe un método aún más sencillo y directo para obtener su propio usuario dentro del portal. El objeto GIS dispone de la propiedad `me`, que devuelve automáticamente el usuario con el que usted ha iniciado sesión. Basta con ejecutar:

```
yo = gis.users.me
yo
```



Dado que está trabajando con su propio usuario, puede consultar y modificar su información personal del mismo modo que lo haría desde la interfaz web en la sección "Editar mi perfil" de ArcGIS Online.

Si tuviera un rol de “Administrador”, además podría modificar cualquier usuario de la organización, así como crearlos, deshabilitarlos o eliminarlos. Un administrador incluso puede automatizar procesos como la lectura de una base de datos o de un archivo Excel con una lista de usuarios y crearlos directamente mediante La API.

A continuación, realice una lectura de algunas propiedades de su usuario e intente modificar alguna de ellas.

- Consulte los créditos disponibles con [la propiedad de availableCredits](#)

```
yo = gis.users.me
yo.availableCredits # Lectura de créditos disponibles
```

```
yo.availableCredits
```

```
79.50166
```

- Actualice la bio del usuario con la propiedad update:

```
yo.update(description="Prueba BIO para el curso de API") #Ejemplo de modificación
```

```
yo
```

```
yo.update(description = "Prueba BIO para el curso de API")
yo
```



yulia streltsova
Bio: Prueba BIO para el curso de API
First Name: yulia
Last Name: streltsova
Username: yulia.streltsova_educacionEsri
Joined: March 02, 2023

- Guarde el Notebook.

4. Perfiles de usuario

Es fundamental comprender que ArcGIS API for Python es un API diseñado para trabajar con su Web GIS. Esto implica que opera bajo un modelo cliente-servidor, donde usted ejecuta tareas desde el lado del cliente mientras el procesamiento y la gestión recaen sobre el backend de ArcGIS Online o Portal for ArcGIS.

A diferencia de ArcPy, que es una librería de Python que se ejecuta dentro de un cliente como ArcGIS Pro o ArcGIS Server, ArcGIS API for Python no funciona como una librería local embebida. En su lugar, actúa como una capa de comunicación que requiere un entorno web en el que exista un portal (ArcGIS Online o Enterprise), ya que todas las operaciones dependen de los privilegios y capacidades del usuario con el que usted se autentique en dicho backend.

En consecuencia, el uso de la API estará siempre condicionado por el perfil de usuario empleado: sus privilegios, sus capacidades administrativas y su nivel de acceso determinarán qué operaciones puede realizar y qué recursos puede consultar dentro de la organización.



Toda la REST API puede consultarse desde el siguiente enlace oficial: [Build powerful apps with ArcGIS services | ArcGIS Developers – REST APIs Documentation](https://developers.arcgis.com/python/guide/build-powerful-apps-with-arcgis-services/).

Esta referencia constituye la base técnica sobre la que se apoyan todas las operaciones que se realizan desde ArcGIS API for Python.

Mediante ArcGIS API for Python, usted puede llevar a cabo una gran variedad de tareas dentro de su Web GIS. Para simplificar y estructurar el tipo de trabajo que puede realizar con este API, hemos clasificado las operaciones en tres perfiles de uso principales:

- Administración
- Creación de contenido
- Análisis espacial

En este capítulo revisaremos cada uno de estos perfiles, explicando sus características, sus capacidades y los escenarios en los que resultan más relevantes.

Administración de tu Web GIS

Dentro de las tareas que un **administrador GIS** puede realizar se encuentra una amplia variedad de acciones fundamentales para la gestión de la organización. Entre ellas destacan:

- **Crear usuarios y roles** en su portal.
- **Clonar portales** o replicar la estructura y el contenido entre entornos.
- **Reasignar contenido** perteneciente a distintos usuarios.
- **Realizar búsquedas** avanzadas sobre el contenido de la organización.
- **Establecer relaciones** entre ítems para estructurar adecuadamente los recursos.
- **Generar informes** detallados sobre la actividad y el estado de los usuarios.

Estas capacidades convierten al administrador en la figura clave para asegurar el correcto funcionamiento del Web GIS, mantener la integridad de los datos y garantizar que los usuarios trabajan dentro de un entorno bien organizado y seguro.

Generando contenidos

Los publicadores de contenidos o aquellos usuarios responsables de generar y mantener información dentro de la organización pueden aprovechar ampliamente las capacidades de ArcGIS API for Python. Entre las tareas más habituales que pueden automatizar o facilitar mediante este API se encuentran:

- **Automatizar la publicación de información**, evitando procesos manuales repetitivos.
- **Actualizar tiles y feature layers** directamente desde sus bases de datos corporativas o fuentes de datos externas.
- **Replicar entornos**, trasladando contenido desde un entorno de desarrollo hacia uno de producción de forma controlada.
- **Inspeccionar y actualizar ítems** para detectar enlaces rotos o referencias a servicios que ya no están disponibles.

Estas funcionalidades permiten mantener la organización actualizada, coherente y alineada con los flujos de trabajo internos, a la vez que reducen la intervención manual y el riesgo de errores.

Ejercicio 5A: Generar una visualización en ArcGIS Notebooks

A la hora de generar una visualización (mapping) en ArcGIS Notebooks mediante ArcGIS API for Python, seguiremos un flujo de trabajo estructurado en tres fases, siendo la tercera de ellas opcional:

- Acceso a los datos
- Análisis y representación
- Exportación o publicación (opcional)



En esta actividad repasaremos el flujo completo. Tanto en el primer paso –acceso a los datos– como en el tercero –exportación–, usted puede trabajar con información almacenada en local o con datos disponibles en la web, ya sea a través de ítems de su portal o de servicios Web GIS.

Paso 1: Analizar los datos mediante diferentes librerías

- Cree un Nuevo notebook para poder realizar este ejercicio y nómbrelo como **"Ejercicio 5A: Generar una visualización en ArcGIS Notebooks"**:
- Ejecute de nuevo la primera celda con la importación de los módulos:

Importar módulos y acceder al portal

```
1]: from arcgis.gis import GIS  
gis = GIS("home")
```

Muchas organizaciones gestionan información mediante ficheros CSV (o Excel). Como se ha visto anteriormente, Python dispone de herramientas como Pandas, capaces de leer fácilmente este tipo de archivos.

- Utilice un fichero CSV que almacena información de terremotos sucedidos en Los Ángeles para realizar una lectura de datos a través de pandas.

Nota: El archivo está disponible públicamente en ArcGIS Online: <https://educacionesri.maps.arcgis.com/sharing/rest/content/items/c0a5c07b85074eeef2b13376314c7b9/data>

- Visualice la tabla csv utilizando variable **terremotos**:

```
[2]: #Lectura del csv a través de pandas
import pandas as pd

[3]: terremotos = pd.read_csv("https://educacionesi.maps.arcgis.com/sharing/rest/content/items/c0a5c07585674eaeaf2b13376314c7b0/data")
terremotos.head()
```

	Unnamed: 0	X	Y	ProgramName	ProjectTitle	DamageExtent	ID
0	0	-118.213772	34.032372	Municipal Facilities R and P	Boyle Heights Sports Center	95	35
1	1	-118.245915	34.045172	Municipal Facilities R and P	Engine Company 23 Arts Center	76	20565
2	2	-118.245911	34.045161	Municipal Facilities R and P	Engine Company 23 Arts Center	76	20566
3	3	-118.407213	34.235234	Bridge Improvement Program	Laurel Canyon Blvd Bridge-1233	78	771
4	4	-118.317632	34.025554	Wastewater - CSP	Wilshire Area System Sww Rehab	58	768

Podrá incluso representar esta información de forma más completa utilizando Matplotlib.

- Importe el módulo matplotlib para poder visualizar los datos gráficamente dentro de una celda utilizando la palabra mágica %matplotlib inline:

```
[4]: #Importar el modulo de matplotlib para poder visualizar los datos graficamente
import matplotlib as plt
```

```
[5]: %matplotlib inline
```

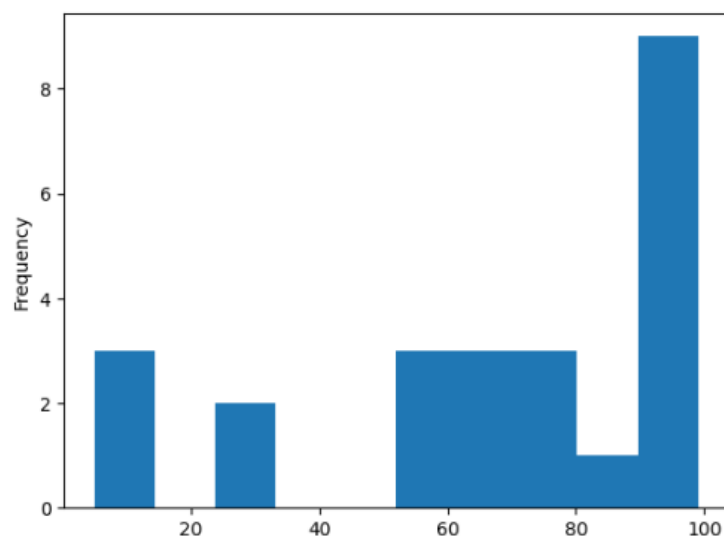
- Ejecute ambas celdas.
- Para poder visualizar los datos gráficamente utilice la columna de visualización "DamageExtent" y seleccione el tipo de gráfico "Histograma" (kind="hist")

```
terremotos["DamageExtent"].plot(kind="hist")
```

- Ejecute la celda.

```
[6]: terremotos["DamageExtent"].plot(kind="hist")
```

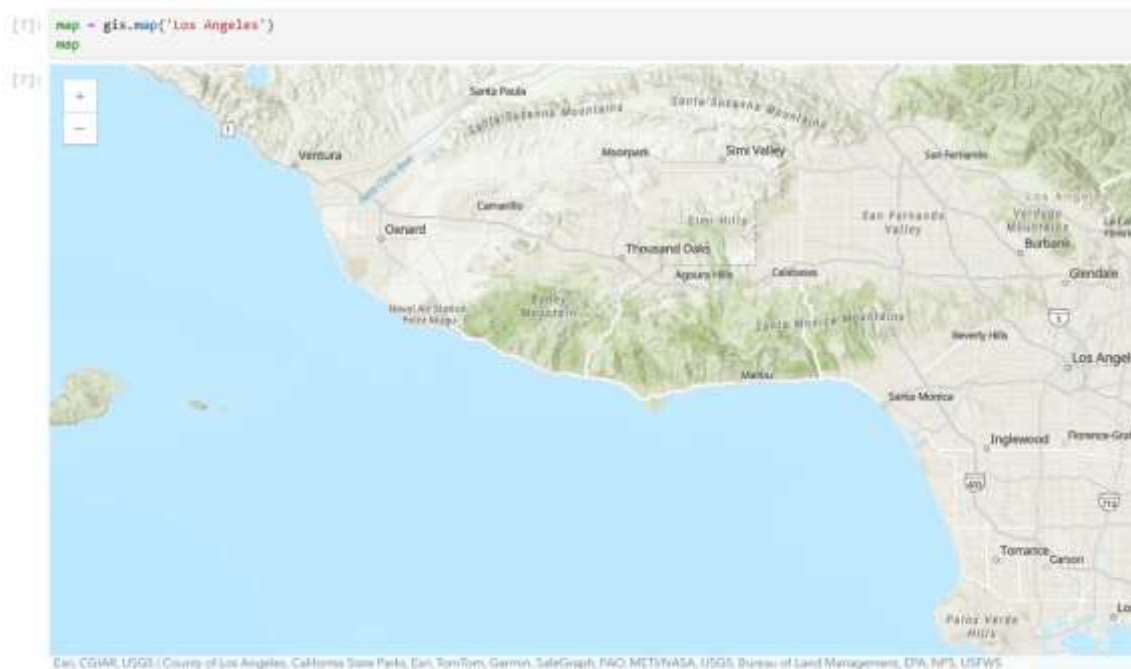
```
[6]: <Axes: ylabel='Frequency'>
```



Desde un punto de vista GIS, aprovecharemos la integración de la API de Python con Pandas para empezar a manipular estos datos con facilidad.

- A continuación, muestre un mapa interactivo ([widget map](#)) centrado en Los Ángeles con el siguiente código:

```
map = gis.map('Los Angeles')
map
```



A continuación, aprovecharemos el módulo spatial que ArcGIS API for Python incorpora a Pandas para convertir el DataFrame de terremotos en un Spatially Enabled DataFrame (SeDF).

- Escriba el siguiente código:

```
sedf = pd.DataFrame.spatial.from_xy(terremotos, "X", "Y")
sedf.head()
```

- Ejecute la celda

The screenshot shows a Jupyter Notebook interface. The first cell contains the code `sedf = pd.DataFrame.spatial.from_xy(terremotos, "X", "Y")` and the second cell contains `sedf.head()`. Below the code is a table showing the first five rows of the resulting SeDF. The table has columns: Unnamed: 0, X, Y, ProgramName, ProjectTitle, DamageExtent, ID, and SHAPE. The SHAPE column contains spatial reference information.

	Unnamed: 0	X	Y	ProgramName	ProjectTitle	DamageExtent	ID	SHAPE
0	0	-118.213772	34.032372	Municipal Facilities R and P	Ecyle Heights Sports Center	95	35	["spatialReference": {"wkid": 4326}, "x": -118...
1	1	-118.245915	34.045172	Municipal Facilities R and P	Engine Company 23 Arts Center	76	20565	["spatialReference": {"wkid": 4326}, "x": -118...
2	2	-118.245911	34.045161	Municipal Facilities R and P	Engine Company 23 Arts Center	76	20566	["spatialReference": {"wkid": 4326}, "x": -118...
3	3	-118.407213	34.235234	Bridge Improvement Program	Laurel Canyon Blvd Bridge-1233	78	771	["spatialReference": {"wkid": 4326}, "x": -118...
4	4	-118.317632	34.025554	Wastewater - CSP	Wilshire Area System Swr Rehab	58	768	["spatialReference": {"wkid": 4326}, "x": -118...

Esta conversión añade el atributo SHAPE, lo que le permitirá trabajar con los datos de manera geográfica y añadirlos al mapa con facilidad.

- Para agregar esos puntos de los terremotos al mapa que se creó anteriormente, primero convierta este DataFrame Espacial al Feature Set:

```
item = sedf.spatial.to_featureset()
```

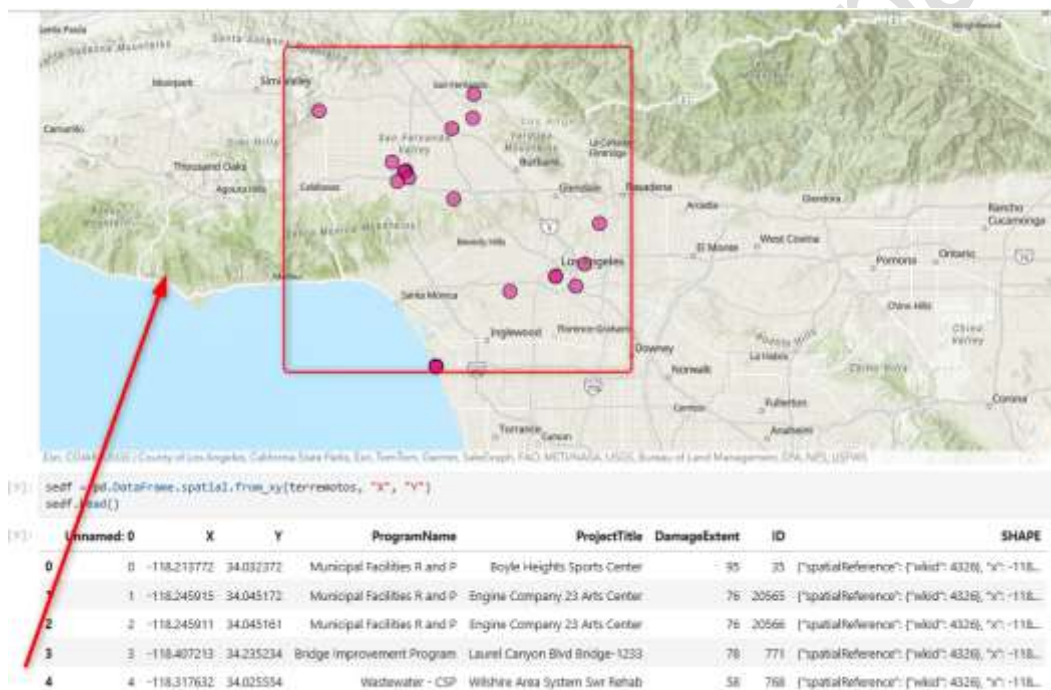
Paso 2: Crear un widget mapa para visualización

- En la siguiente línea insierte el [método add dentro de la clase content](#) para poder agregar la capa recién creada al widget mapa:

```
map.content.add(item)
```

```
[11]: item = sedf.spatial.to_featureset()  
      map.content.add(item)
```

- Para poder visualizar los puntos correctamente vuelva a visualizar el widget mapa:



Una vez creado el mapa, usted puede modificar diversas propiedades, tales como el mapa base (consulte la propiedad `basemaps`), el nivel de zoom o el centrado del mapa, ajustando así la visualización a las necesidades de su análisis.

Paso 3: Actualizar propiedades del mapa

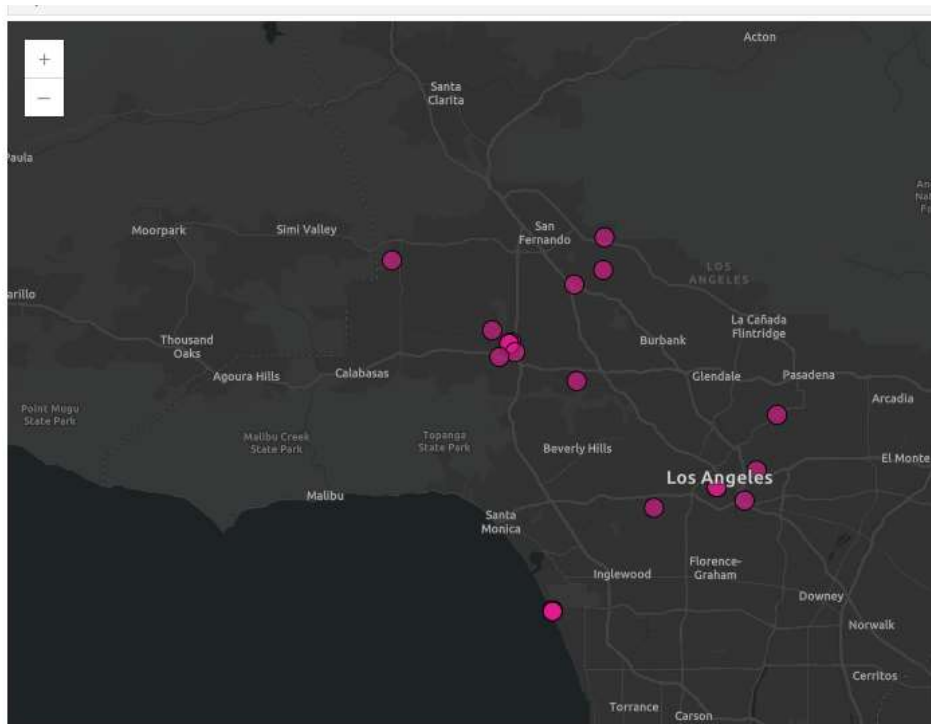
- Actualice un mapa base del mapa actual utilizando el atributo anidado `map.basemap.basemap`, asignándole el nombre del estilo deseado:

```
map.basemap.basemap = "dark-gray-vector"
```

Actualizar propiedades del mapa

```
[17]: map.basemap.basemap = "dark-gray-vector"
```

- Ejecute la celda y observe los resultados obtenidos sobre el widget mapa:



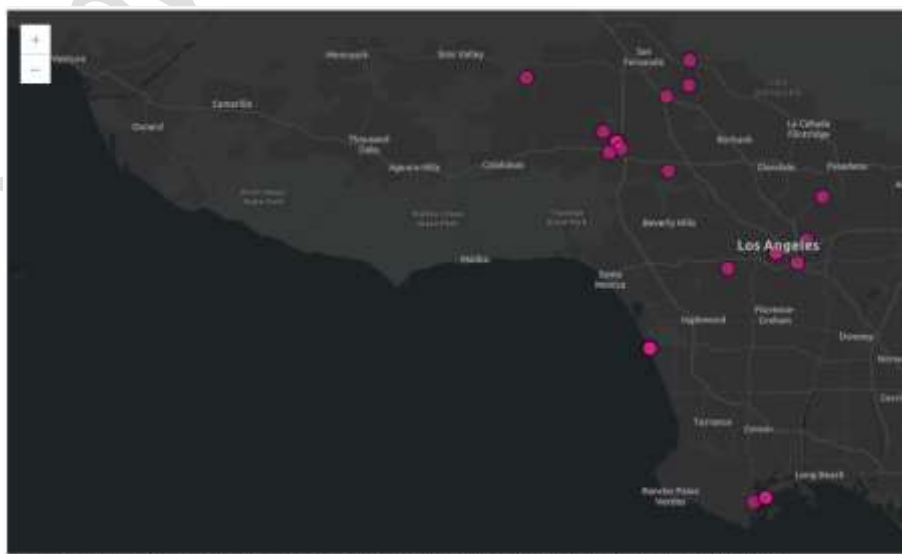
Nota: Lea más ejemplo en la página oficial de las APIs para entender mejor la sintaxis del uso de widget mapa: [Using the map widget | ArcGIS API for Python | Esri Developer](#)

- Establezca el centro del mapa con coordenadas conocidas:

```
map.center = [34, -118]
```

```
[20]: #Establecer un centro del mapa con coordenadas conocidas  
map.center = [34, -118]
```

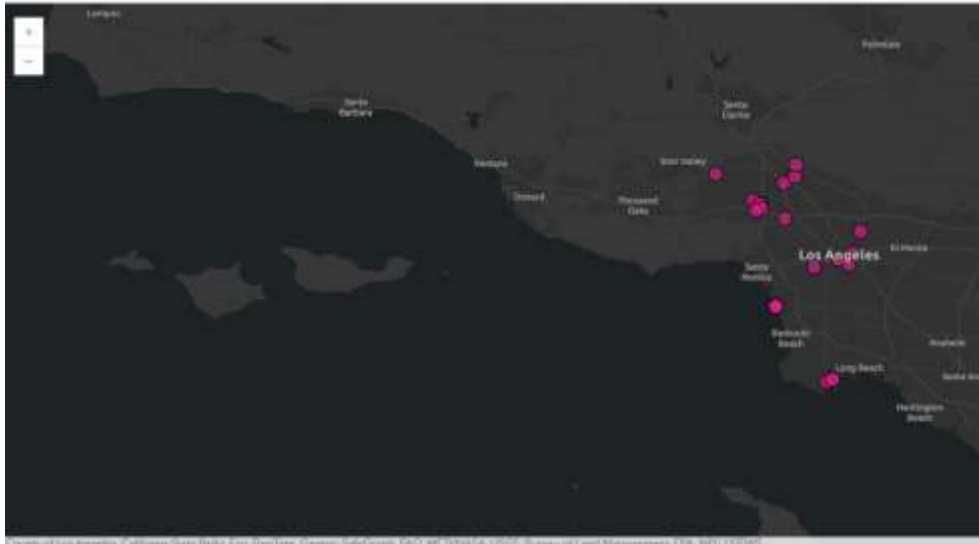
El mapa se actualiza de forma inmediata, centrando la vista en las coordenadas indicadas.



Asimismo, es posible **establecer diferentes niveles de zoom** en el mapa.

- Para fijar manualmente un nivel concreto, utilice la propiedad zoom:

```
map.zoom = 8
```



Por otro lado, el widget permite ajustar la escala de manera automática para que el mapa **encuadre una capa específica**.

- Para ello, emplee el método zoom_to_layer():

```
map.zoom_to_layer(sedf)
```



En ambos casos, el mapa se actualizará inmediatamente reflejando el nuevo nivel de detalle.

Paso 4: Aplicar la simbología smart mapping

La visualización de capas puede mejorarse significativamente mediante el uso de los renderers disponibles en La API de ArcGIS para JavaScript, los cuales permiten aplicar simbologías avanzadas directamente desde el entorno de ArcGIS Notebooks.

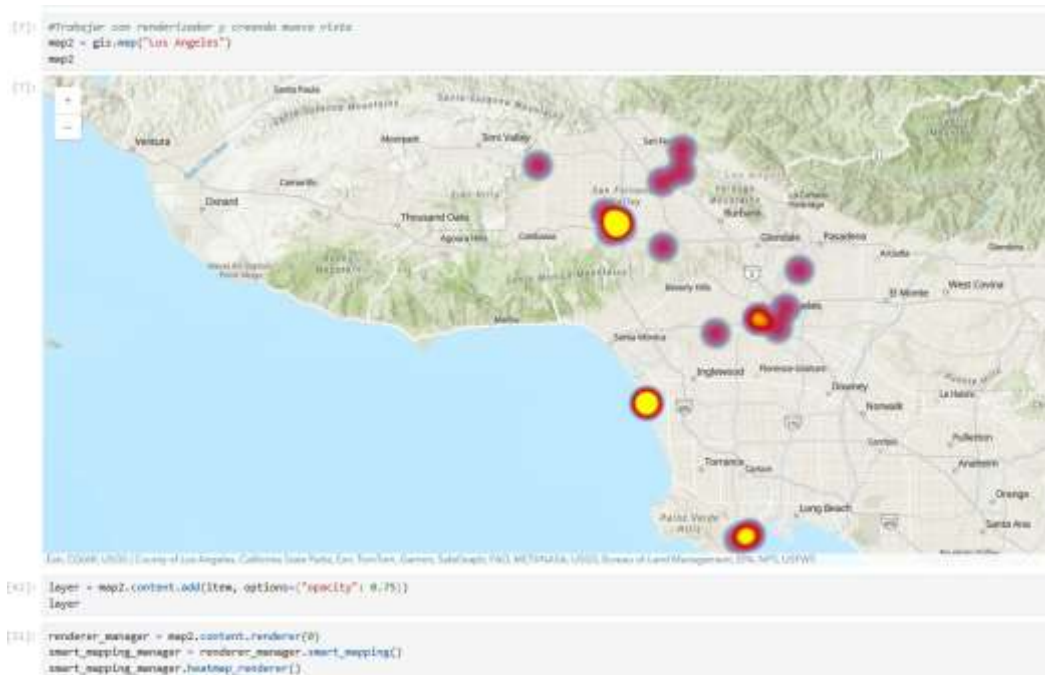
Nota: ver más en detalle el [uso de renderizadores](#) y los ejemplos reales: [Smart Mapping | ArcGIS API for Python | Esri Developer](#)

- Cree una nueva celda para crear nueva vista para poder visualizar posteriormente un renderizador del mapa de calor:

```
#Trabajar con renderizador y creando nueva vista
```

```
map2 = gis.map("Los Angeles")
```

```
map2
```



Por último, el widget de mapa puede transformarse directamente en un Web Map dentro de su organización, permitiendo su almacenamiento, reutilización y posterior uso en aplicaciones y visores web.

- Escriba el siguiente código utilizando el método .save() para poder guardar el mapa web con los metadatos propuestos:

```
#Guardar el widget mapa como WebMap
```

```
map.save({'title': 'Terremotos LA_YS', 'snippet': 'Terremotos en Los Angeles con daño mayor que 60', 'tags': 'terremotos, notebook'})
```



Al ejecutar la celda se aparece la miniatura del Mapa Web y es clickable para poder consultar los metadatos, abrir detalles y abrirlo posteriormente en MapViewer.

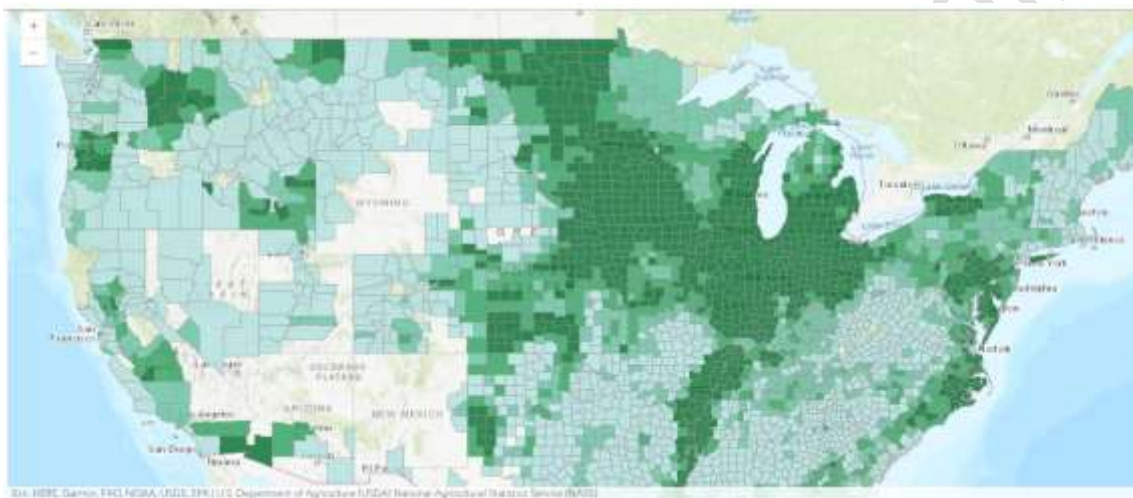
- Guarde el Notebook.

Ejercicio 5B: Smart Mapping con personalización del renderizador

Nota: Para ver los ejemplos de Smart Mapping dirige a la página de Samples de ArcGIS API for Python: [Smart Mapping | ArcGIS API for Python | Esri Developer](#)

Aunque los métodos de smart mapping eligen automáticamente los saltos de clase, también podemos usar métodos y clases que nos permitan definir los saltos de clase y simbología. Para definir la simbología del renderizador podemos usar la SDK de JavaScript dentro de la API de Python.

En este ejercicio vamos a representar el porcentaje de superficie destinada a cultivos en Estados Unidos. El resultado que obtendremos será el siguiente:



Paso 1: Trabajar con el renderizador básico de Smart mapping

- Cree un Nuevo notebook para poder realizar este ejercicio y nómbrelo como **"Ejercicio 5B: Personalización del renderizador"**:
- Importe el módulo GIS y conecte con el portal actual:

Importar los módulos necesarios

```
from arcgis.gis import GIS
gis = GIS("home")
```

Nota: En este ejercicio se trabajará con el servicio https://services.arcgis.com/V6ZHFr6zdgNZuVG0/arcgis/rest/services/USA_County_Crops_2007/FeatureServer/0

- Para trabajar con servicio web importe la clase Service desde el módulo arcgis.layers:

```
from arcgis.layers import Service
```

La clase `Service` permite a los usuarios pasar una cadena `url` o un [Elemento](#), junto con una conexión [SIG](#) opcional o un objeto [Servidor](#) específico para devolver una instancia del objeto específico de ArcGIS API for Python que representa el servicio.

- Declara el servicio como variable:

```
usa_flayer = Service("https://services.arcgis.com/V6ZHFr6zdgNZuVG0/arcgis/rest/services/USA_County_Crops_2007/FeatureServer/0")
```

- Cree el widget mapa y agregue el servicio como una capa:

```
map1 = gis.map("Midwest")
map1.content.add(usa_flayer)
map1
```

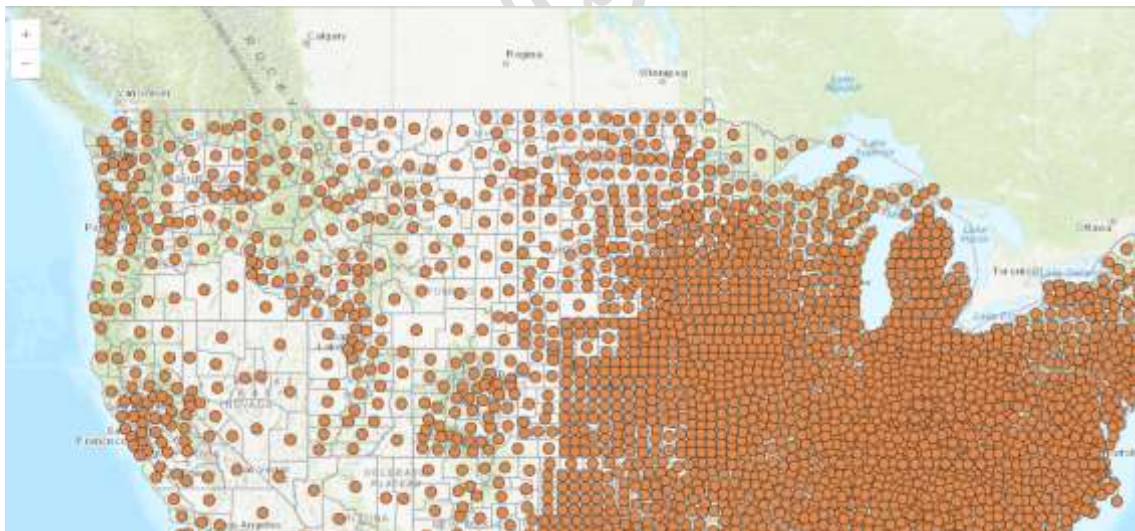
- Compruebe el siguiente código:

```
from arcgis.layers import Service

usa_flayer = Service(
    "https://services.arcgis.com/V6ZHFr6zdgNZuVG0/arcgis/rest/services/USA_County_Crops_2007/FeatureServer/0"
)

map1 = gis.map("Midwest")
map1.content.add(usa_flayer)
map1
```

El resultado de agregación del servicio:

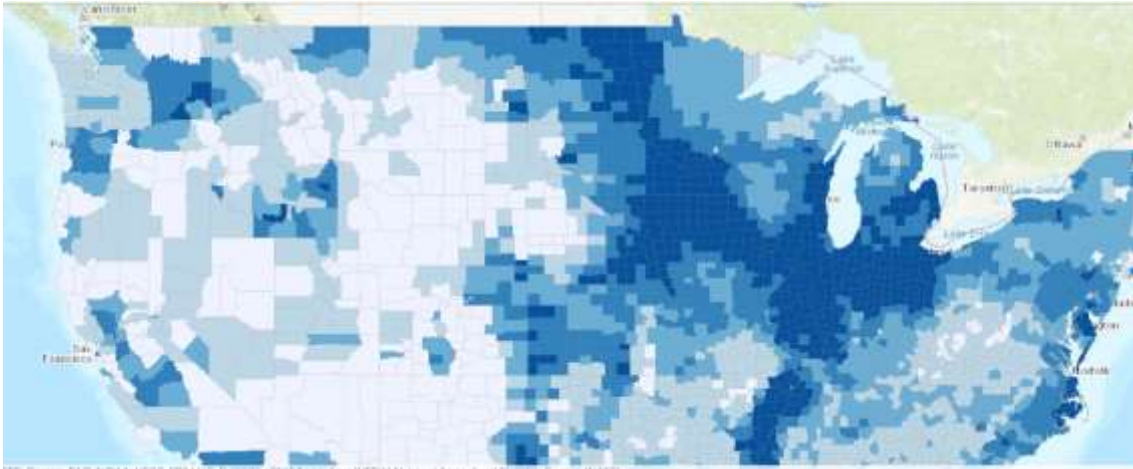


- Aplica el renderer básico según los ejemplos de Samples de Smart Mapping, teniendo en cuenta que el campo que nos interesa visualizar es `M086_07` que corresponde a *Las acres de terreno cultivable total como porcentaje de la superficie del terreno en acres*:

```
renderer_manager = map1.content.renderer(0)
smart_mapping_manager = renderer_manager.smart_mapping()
```

```
smart_mapping_manager.class_breaks_renderer(break_type="color",
field="M086_07")
```

- Ejecute la celda para observar el resultado final



Paso 2: Personalizar el renderizador

El objetivo principal de este ejercicio es definir nosotros toda la representación. En este caso se trata de un class break renderer por lo que deberemos definir los valores a partir de los cuales pasaremos de una clase a otra. El campo en cuestión es el M086_07.

- Cargue las clases necesarias para esta representación:

```
from arcgis.map.renderers import (
    ClassBreaksRenderer,
    ClassBreakInfo,
    SizeInfoVisualVariable,
)
from arcgis.map.symbols import SimpleLineSymbolEsriSLS,
SimpleFillSymbolEsriSFS
```

- Cree el widget mapa 2 para poder visualizar la capa con otro renderizador:

```
#crear Widget Mapa 2
map2 = gis.map("Midwest")
usa_flayer = Service(
    "https://services.arcgis.com/V6ZHFr6zdgNHuVGB/arcgis/rest/services/USA_County_Crops_2007/FeatureServer/0"
)
```

Antes de definir el renderizador deberemos definir las clases con ClassBreakInfo donde deberemos especificar el valor máximo de la clase, la etiqueta y la simbología que queremos que se aplique.

- Especifique la simbología de las clases declarando la variable como class_break_infos:

```

class_break_infos = [
    ClassBreakInfo(
        class_max_value=10,
        label="< 10%",
        symbol=SimpleFillSymbolEsriSFS(
            **{
                "type": "esriSFS",
                "style": "esriSFSSolid",
                "color": [237, 248, 251, 204], # Light greenish color
                "outline": {"color": [153, 153, 153, 255], "width":
0.375},
            }
        ),
    ),
    ClassBreakInfo(
        class_max_value=39.5,
        label="10% - 39.5%",
        symbol=SimpleFillSymbolEsriSFS(
            **{
                "type": "esriSFS",
                "style": "esriSFSSolid",
                "color": [178, 226, 226, 204], # Slightly darker green
                "outline": {"color": [153, 153, 153, 255], "width":
0.375},
            }
        ),
    ),
    ClassBreakInfo(
        class_max_value=54,
        label="39.5% - 54%",
        symbol=SimpleFillSymbolEsriSFS(
            **{
                "type": "esriSFS",

```

```

        "style": "esriSFSSolid",
        "color": [102, 194, 164, 204], # Medium green
        "outline": {"color": [153, 153, 153, 255], "width":
0.375},
    }
),
),
ClassBreakInfo(
    class_max_value=68,
    label="54% - 68%",
    symbol=SimpleFillSymbolEsriSFS(
        **{
            "type": "esriSFS",
            "style": "esriSFSSolid",
            "color": [44, 162, 95, 204], # Dark green
            "outline": {"color": [153, 153, 153, 255], "width":
0.375},
        }
    ),
),
ClassBreakInfo(
    class_max_value=100,
    label="> 68%",
    symbol=SimpleFillSymbolEsriSFS(
        **{
            "type": "esriSFS",
            "style": "esriSFSSolid",
            "color": [0, 109, 44, 204], # Darkest green
            "outline": {"color": [153, 153, 153, 255], "width":
0.375},
        }
    ),
),
),

```

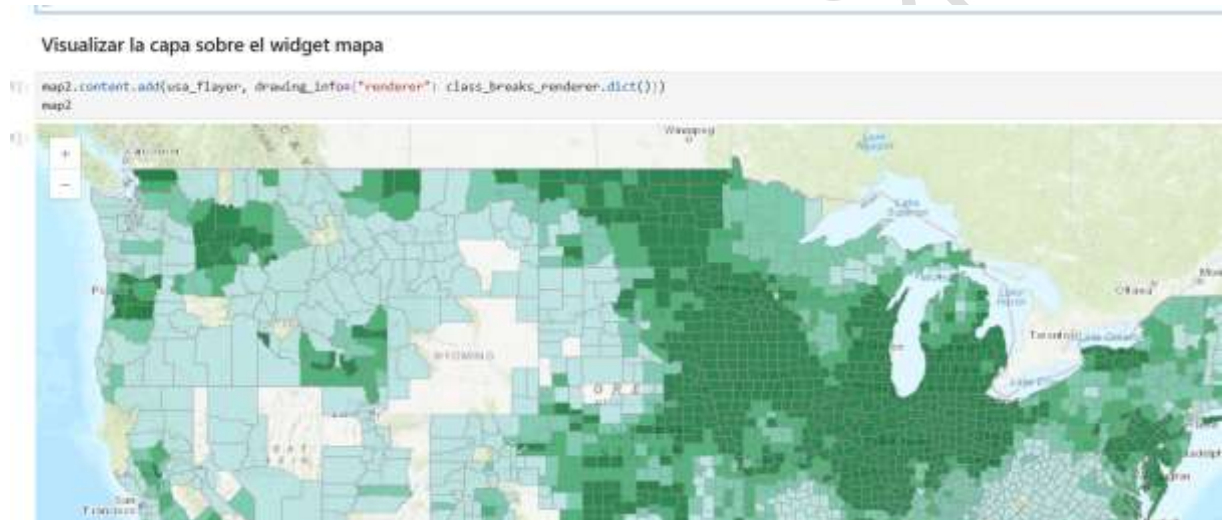
]

- Una vez definidos los límites entre clases, pasamos el parámetro al renderizador indicando el campo que vamos a representar.

#Definir el renderizador

```
class_breaks_renderer = ClassBreaksRenderer(  
    field="M086_07",  
    class_break_infos=class_break_infos,  
    normalization_type=None, # Cambie a «esriNormalizeByField» si desea  
    normalizar.  
)
```

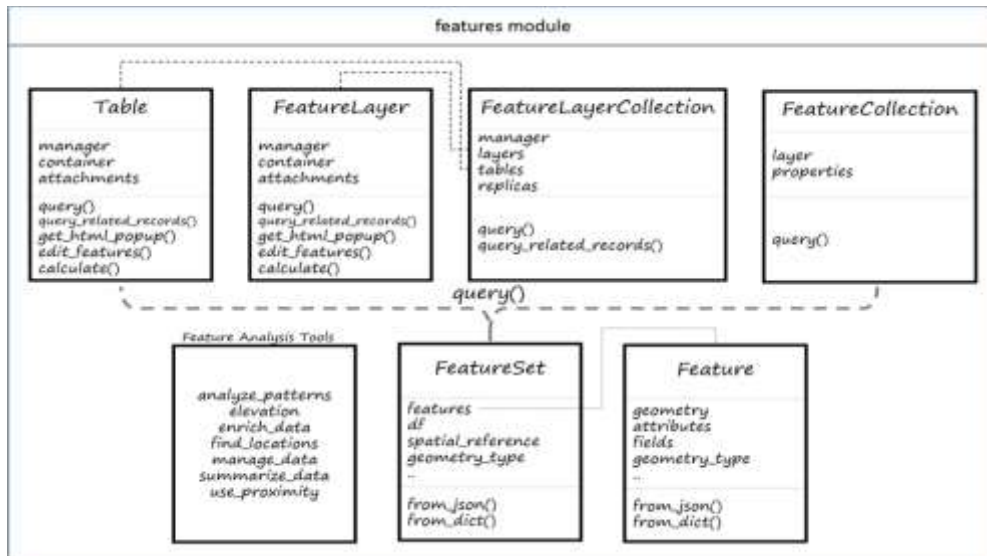
- Por último, agregue la capa con el método `.content.add`, indicando el renderizador que acabamos de crear:



- Guarde el Notebook para no perder los cambios.

5. Módulo Features

El módulo **Features** permite trabajar directamente con entidades geográficas. Una *feature* o entidad representa la unidad mínima de información en un sistema GIS (punto, línea o polígono) dentro de una clase de entidad, del mismo modo que una fila (row) constituye un registro dentro de una tabla.



Este módulo reúne las clases necesarias para gestionar entidades, clases de entidad, tablas, colecciones y subconjuntos de datos. Las clases principales incorporan, además, métodos estáticos que facilitan el acceso a objetos de distintas clases, lo que permite alcanzar un mismo resultado a través de diferentes enfoques. Este diseño reduce la complejidad para el programador y agiliza los flujos de trabajo.

En el notebook correspondiente, revise el significado y finalidad de los componentes principales del módulo:

- FeatureLayerCollection
- FeatureLayer
- FeatureCollection
- FeatureSet
- Feature

A continuación, puede continuar trabajando con la capa de terremotos que previamente se ha guardado en su organización como un ítem.

Dicho ítem se corresponde con un mapa Web.

Nota: Le recomendamos acceder a la URL del servicio para inspeccionar su estructura y verificar los datos publicados.

➤ Acceda al ítem usando la propiedad `content` con el método `search()`:

```
terremotosItems = gis.content.search('Terremotos', max_items=20)
terremotosItems
```

Acceder a la capa de terremotos

```
[27]: terremotosItems = gis.content.search('Terremotos', max_items=
      terremotosItems

[27]: [<Item title:"Terremotos LA_Tutor" type:Web Map owner:Tutor_M
```

Ejercicio 6: Consultando datos

Es posible realizar consultas sobre datos publicados como servicios en ArcGIS Online y ArcGIS Enterprise. En este ejercicio se llevarán a cabo consultas sobre este tipo de servicios para familiarizarse con su funcionamiento.

Trabajaremos con un fichero CSV que contiene información sobre las estaciones de servicio que dispensan biodiésel en España. Para simplificar el proceso, dicho fichero ha sido previamente publicado en ArcGIS Online y está disponible en el siguiente enlace: <https://arcg.is/0ufOqr>

Nota: Como se observa en el propio ítem, se proporciona acceso al Feature Service subyacente, que será el recurso utilizado para acceder a los datos durante este ejercicio:

https://services5.arcgis.com/ziEoKDcukFDiV7qZ/arcgis/rest/services/Estaciones_de_servicio_biodiesel/FeatureServer/0

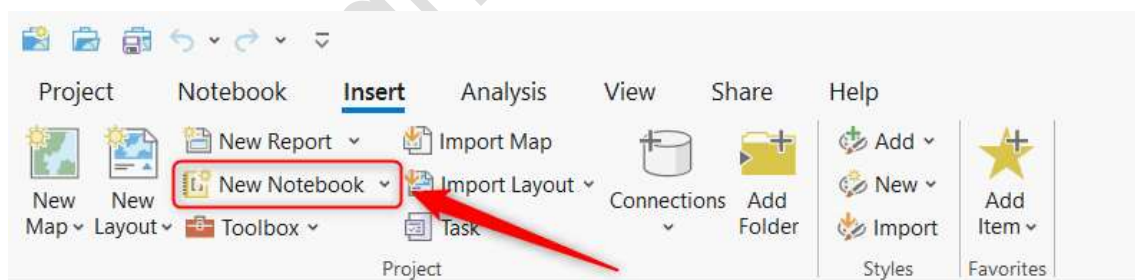
Este ejercicio ha sido realizado en el entorno local dentro de Notebooks de ArcGIS Pro, pero también se puede realizar dentro del entorno Web en ArcGIS Notebooks.

Paso 1: Acceso al portal GIS

En este ejercicio trabajaremos en el entorno de ArcGIS Pro.

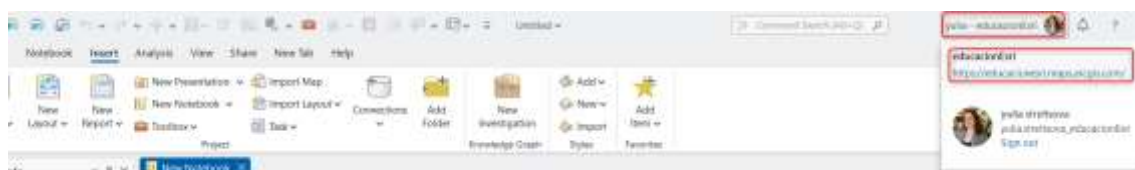
- Para crear un Notebook desde ArcGIS Pro, diríjase a la pestaña Insertar y, dentro de la sección Project, haga clic en New Notebook.

Nota: La creación de un proyecto para realizar estas tareas no es obligatoria, aunque sí recomendable. También es posible comenzar a trabajar sin utilizar una plantilla y guardar el proyecto posteriormente. La creación y el guardado de proyectos no se abordan en este ejercicio, puesto que estos procedimientos ya se trataron en módulos anteriores.



- Una vez creado el Notebook, proceda a establecer la conexión con el portal de su organización.

Nota: Asegúrese de que ArcGIS Pro esté conectado al portal de la educación: (<https://educacionesri.maps.arcgis.com/>) antes de continuar.



Para conectar con el portal GIS, debe importar previamente el módulo gis.

- Para ello, añade en la celda del Notebook la siguiente instrucción:

```
[2]: from arcgis.gis import GIS
```

Nota: Para ejecutar la celda haga clic en **Run** (o usa el teclado **Shift + Enter**)

Para crear el objeto GIS, debería proporcionar la URL del portal y sus credenciales de acceso, tal y como se especifica en el constructor de la clase GIS ([véase arcgis.gis module | ArcGIS API for Python](#)).

No obstante, dado que está trabajando dentro de ArcGIS Pro y ya se encuentra conectado al portal de su organización, no es necesario introducir la URL ni las credenciales. En este caso, únicamente debe indicar "home" como argumento en la clase GIS, lo que permitirá utilizar directamente la conexión activa.

- Escriba el siguiente código en la próxima celda de Notebook:

```
[2]: gis=GIS("Home")
      gis
```

```
[2]: GIS @ https://educacionesri.maps.arcgis.com/
```

A continuación, vamos a localizar el Feature Layer utilizando la URL que se ha guardado previamente. Para ello, emplearemos directamente el módulo `arcgis.features` de la API, tal y como se describe en la documentación oficial ([arcgis.features module | ArcGIS API for Python](#)).

- Importe el módulo `features` con varias clases: `FeatureLayer`, `Feature`, `FeatureSet`.

```
[4]: from arcgis.features import FeatureLayer
      from arcgis.features import Feature, FeatureSet
```

- Cree un objeto de tipo **FeatureLayer** a partir de la URL del servicio publicada en ArcGIS Online:

```
[5]: estaciones = FeatureLayer("https://services5.arcgis.com/ziEoKcukFDiV7qZ/arcgis/rest/services/Estaciones_de_servicio_biodiesel/FeatureLayer/0")
[5]: <FeatureLayer url="https://services5.arcgis.com/ziEoKcukFDiV7qZ/arcgis/rest/services/Estaciones_de_servicio_biodiesel/FeatureLayer/0">
```

La clase `FeatureLayer` actúa como un enlace directo al *Feature Service*, permitiéndole consultar, filtrar y manipular los datos expuestos por dicho servicio.

Paso 2: Consulta de los datos

Antes de comenzar a realizar consultas, resulta útil revisar algunas de las características principales de la capa. En muchas ocasiones no se dispone de un conocimiento detallado sobre los datos que se van a analizar, y Python junto con Jupyter ofrece herramientas muy eficaces para explorarlos de forma rápida.

Por ejemplo, es posible consultar las capacidades del servicio y los campos disponibles mediante [las propiedades del objeto FeatureLayer](#).

- Consulte la propiedad de capacidades y campos mediante el siguiente código:

```
estaciones.properties.capabilities
estaciones.properties.fields
```

```
[6]: estaciones.properties.capabilities
```

```
[6]: 'Query'
```

```
[7]: estaciones.properties.fields
```

```
[7]: [{
  "name": "F__OBJECTID",
  "type": "esriFieldTypeInteger",
  "actualType": "int",
  "alias": "__OBJECTID",
  "sqlType": "sqlTypeInteger",
  "nullable": true,
  "editable": true,
  "domain": null,
  "defaultValue": null
},
{
  "name": "X",
  "type": "esriFieldTypeDouble",
  "actualType": "float",
  "alias": "X",
  "sqlType": "sqlTypeFloat",
  "nullable": true,
  "editable": true,
  "domain": null,
  "defaultValue": null
},
{
  "name": "Y",
  "type": "esriFieldTypeDouble",
  "actualType": "float",
  "alias": "Y",
  "sqlType": "sqlTypeFloat",
  "nullable": true,
  "editable": true,
  "domain": null,
  "defaultValue": null
},
}]
```

El resultado mostrado corresponde a **una lista de diccionarios**. Cada elemento de la lista es un diccionario que describe un campo del *Feature Layer*, incluyendo propiedades como:

- name → nombre interno del campo
- type → tipo de dato según la especificación de ArcGIS (por ejemplo, esriFieldTypeDouble, esriFieldTypeString)
- actualType → tipo de dato real en la base de datos
- alias → nombre visible del campo

- nullable → si admite valores nulos
- editable → si puede modificarse
- length → longitud del campo en campos de texto
- defaultValue → valor predeterminado, si existe

Puede comprobar de forma más práctica qué campos contiene la capa y qué tipo de datos almacena cada uno de ellos.

- Construya el iterador for para recorrer la lista e imprimir el nombre y el tipo de campo:

```
[8]: for field in estaciones.properties.fields:
    print(f"{field.name} - {field.type}")

F_OBJECTID - esriFieldTypeInteger
X - esriFieldTypeDouble
Y - esriFieldTypeDouble
Estacion - esriFieldTypeString
Precio - esriFieldTypeInteger
x2 - esriFieldTypeDouble
y2 - esriFieldTypeDouble
ObjectId - esriFieldTypeOID
```

Es el momento de realizar consultas utilizando el método query. En este ejercicio recuperaremos aquellas estaciones de servicio que ofrecen el producto a un precio superior a 1,280 céntimos de euro y, por otro lado, aquellas cuyo precio es inferior a dicha cantidad.

Con estos dos subconjuntos de datos, crearemos un mapa que mostrará ambas capas diferenciadas: una correspondiente al combustible más caro y otra al más económico.

- Cree 2 variables con las consultas query ("Precio >=1280" y "Precio < 1280")

```
1 resultCaros = estaciones.query("Precio >= 1280")
2 resultBaratos = estaciones.query("Precio < 1280")
```

Existen un gran volumen de opciones en el método query que nos aportan la capacidad de hacer cualquier tipo de consulta:



Signature

```
estaciones.query(
    where: 'str' = '1=1',
    out_fields: 'Union[str, list[str]]' = '*',
    time_filter: 'Optional[list[datetime]]' = None,
    geometry_filter: 'Optional[GeometryFilter]' = None,
    return_geometry: 'bool' = True,
    return_count_only: 'bool' = False,
    return_ids_only: 'bool' = False,
    return_distinct_values: 'bool' = False,
    return_extent_only: 'bool' = False,
    group_by_fields_for_statistics: 'Optional[str]' = None,
    statistic_filter: 'Optional[StatisticFilter]' = None,
    result_offset: 'Optional[int]' = None,
    result_record_count: 'Optional[int]' = None,
    object_ids: 'Optional[list[str]]' = None,
    distinct: 'Optional[list[str]]' = None
)
```

```
1 resultCaros = estaciones.query("Precio >= 1280")
2 resultBaratos = estaciones.query("Precio < 1280")
```

- Obtenga un recuento con la opción `return_count_only = True`:

```
[10]: estaciones.query(where='Precio >= 1280', return_count_only=True)
[10]: 191
[11]: estaciones.query(where='Precio < 1280', return_count_only=True)
[11]: 174
```

Paso 3: Mapa que muestre los resultados de la consulta

Por último, cree un mapa centrado en España y añada a dicho mapa el resultado de las dos consultas realizadas.

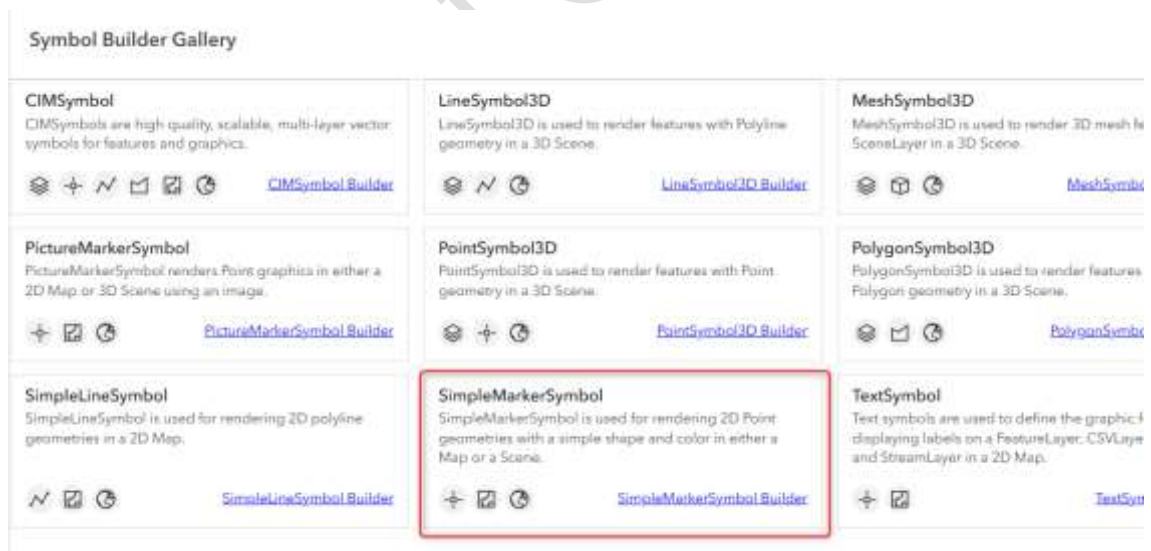
El resultado devuelto por una consulta es un FeatureSet, es decir, un conjunto de entidades compuesto por sus geometrías y sus atributos. Para poder representar estas entidades en un mapa, es necesario asignar una simbología adecuada.

En este sentido, podemos afirmar que cualquier elemento gráfico mostrado en un mapa debe disponer, como mínimo, de dos componentes esenciales:

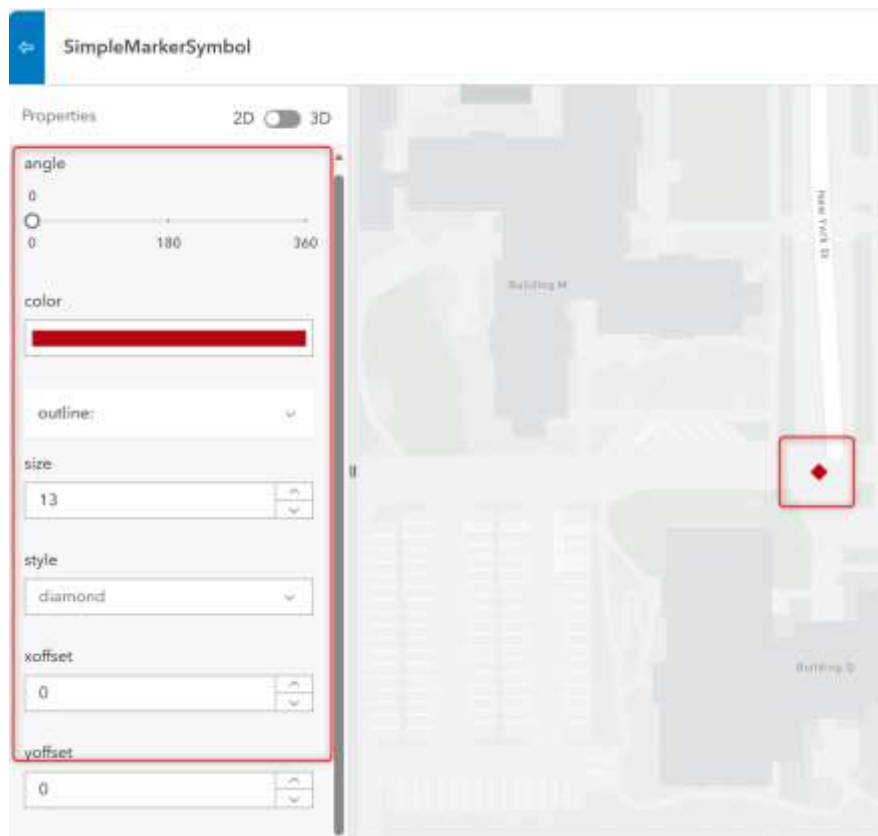
- Una geometría (punto, línea o polígono), que define su localización y forma.
- Una simbología, que determina cómo se representará visualmente dicha geometría en el mapa.

Nota: No es sencillo describir simbologías, pero contamos con una herramienta gráfica que nos puede ayudar: [Symbol Builder \(sagewall.github.io\)](https://sagewall.github.io)

➤ En la galería de símbolos seleccione SimpleMarkerSymbol:



➤ Seleccione un símbolo a su gusto desde el panel izquierdo para gasolinera caras:



- Copie el código de JSON de este símbolo y cree una variable de symbolCaro:

```

ESM  CDN  JSON
JSON
Copy Snippet

import SimpleMarkerSymbol from "@arcgis/core/symbols"
const simpleMarkerSymbol = SimpleMarkerSymbol.fromJSON({
  "type": "esriSMS",
  "color": [194, 0, 10, 255],
  "angle": 0,
  "xoffset": 0,
  "yoffset": 0,
  "size": 13,
  "style": "esriSMSDiamond",
  "outline": {
    "type": "esriSLS",
    "color": [252, 252, 252, 255],
    "width": 1,
    "style": "esriSLSolid"
  }
})

```

```

symbolCaro = {
  "type": "esriSMS",
  "color": [
    194,
    0,
    10,
    255
  ],
  "angle": 0,
  "xoffset": 0,
  "yoffset": 0,
  "size": 13,
  "style": "esriMSDiamond",
  "outline": {
    "type": "esriSLS",
    "color": [
      252,
      252,
      252,
      255
    ],
    "width": 1,
    "style": "esriSLSolid"
  }
}

```

- Repita los pasos para construir el símbolo para gasolineras baratas y copie JSON como valor de la nueva variable:

```

[17]: symbolBarato = {
  "type": "esriSMS",
  "color": [
    23,
    0,
    194,
    255
  ],
  "angle": 0,
  "xoffset": 0,
  "yoffset": 0,
  "size": 12,
  "style": "esriSMSSquare",
  "outline": {
    "type": "esriSLS",
    "color": [
      252,
      252,
      252,
      255
    ],
    "width": 1,
    "style": "esriSLSolid"
  }
}

```

Para visualizar las dos categorías de estaciones de servicio (precio alto y precio bajo), es necesario convertir los resultados de las consultas en objetos FeatureSet. A continuación, se asigna una simbología distinta a cada conjunto mediante SimpleRenderer y se añaden ambos al mapa utilizando map.content.add(). Este enfoque permite representar simultáneamente diferentes subconjuntos de datos con estilos diferenciados en un único mapa.

Convierte cada consulta en un FeatureSet con el siguiente código:

```

from arcgis.features import FeatureSet, Feature

def to_fs(result):
    features = []
    for f in result.features:
        geom = {
            "x": f.attributes["X"],
            "y": f.attributes["Y"],
            "spatialReference": {"wkid": 4326}
        }
        features.append(Feature(geometry=geom,
                                attributes=f.attributes))
    return FeatureSet(features)

fs_caros = to_fs(resultCaros)
fs_baratos = to_fs(resultBaratos)

```

```

[19]: from arcgis.features import FeatureSet, Feature

def to_fs(result):
    features = []
    for f in result.features:
        geom = {
            "x": f.attributes["X"],
            "y": f.attributes["Y"],
            "spatialReference": {"wkid": 4326}
        }
        features.append(Feature(geometry=geom, attributes=f.attributes))
    return FeatureSet(features)

fs_caros = to_fs(resultCaros)
fs_baratos = to_fs(resultBaratos)

```

Creación de dos simbologías distintas en Runtime 12.X se hace mediante el uso de `renderer` (simbología) para un conjunto de entidades.

- `renderer_caros` → simbología para gasolineras caras
- `renderer_barato` → simbología para gasolineras baratas

➤ Escriba este código para crear 2 `renderers`:

```
from arcgis.map.renderers import SimpleRenderer

renderer_caro = SimpleRenderer(symbol=symbolCaro)
renderer_barato = SimpleRenderer(symbol=symbolBarato)
```

donde *symbolBarato* lo define igual que *symbolCaro* pero con otro color/forma)

```
[20]: from arcgis.map.renderers import SimpleRenderer

      renderer_caro = SimpleRenderer(symbol=symbolCaro)
      renderer_barato = SimpleRenderer(symbol=symbolBarato)
```

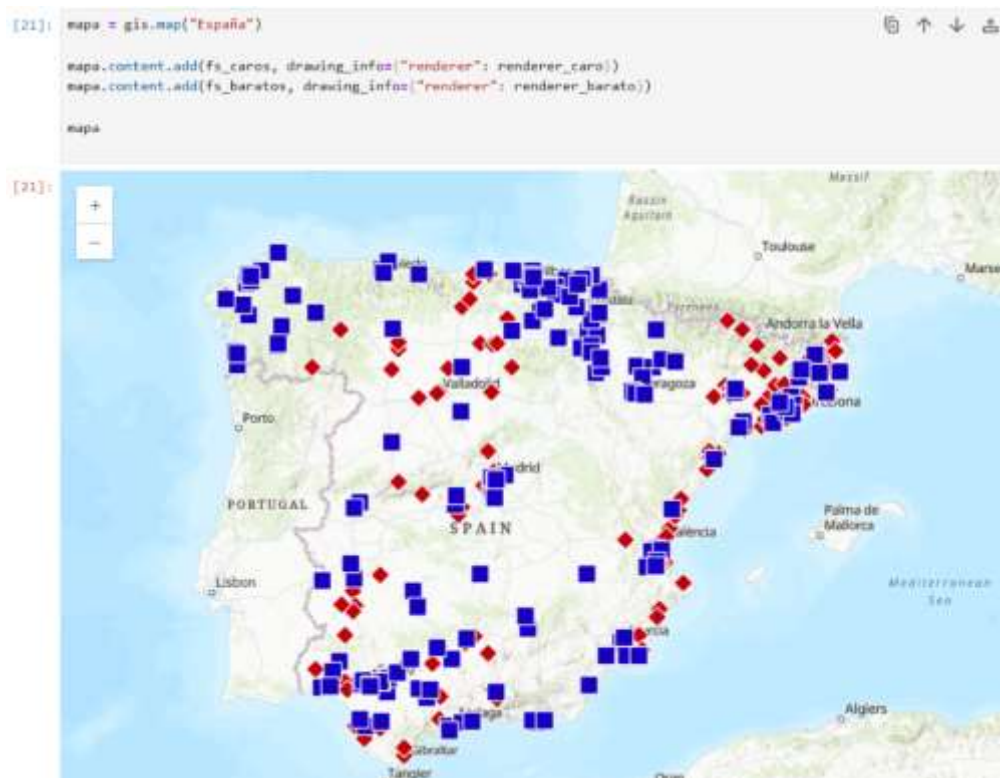
Para poder agregar estas simbologías al mapa, utilizaremos el flujo de trabajo descrito anteriormente. En primer lugar, debemos crear una vista de mapa mediante la generación del widget de mapa utilizando el método `gis.map()` y posteriormente se agregarán ambos `FeatureSets` creados como variables `fs_caros` y `fs_baratos`.

➤ Escriba el siguiente código para poder agregar dos `FeatureSet` a la vista creada:

```
mapa = gis.map("España")

mapa.content.add(fs_caros, drawing_info={"renderer": renderer_caro})
mapa.content.add(fs_baratos, drawing_info={"renderer":
renderer_barato})

mapa
```



- Guarde el widget mapa como un Mapa Web con la metodología que se ha aprendido anteriormente con el método .save() teniendo en cuenta esto metadatos:

```
{'title': 'Gasolineras en España_<iniciales>', 'snippet': 'Distribución de las gasolineras en España y sus clasificaciones dividos en 2 grupos_caras y baratas', 'tags': 'gasolineras, España'}
```



- Renombre el Notebook como **"Ejercicio6_Consultando datos.ipynb"**
- Guarda el Proyecto como **Notebooks_API.aprx** sobre la ruta corta (como ejemplo C://EsriTraining/PAPI).

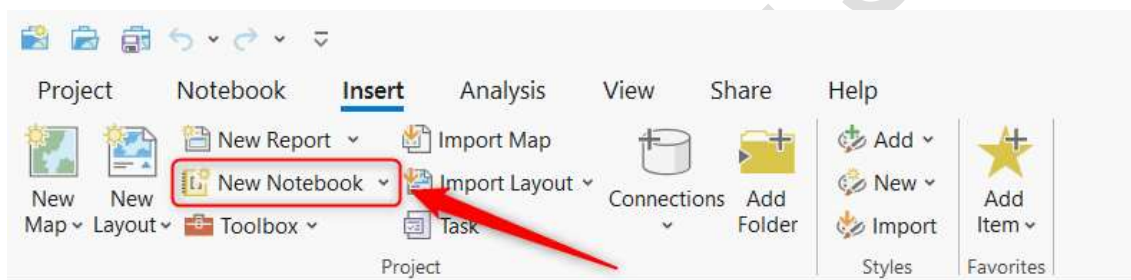
Ejercicio 7: Análisis de los avisos recibidos en el distrito de Moratalaz

Para este ejercicio utilizaremos los avisos registrados por el Ayuntamiento de Madrid durante el año 2023. La fuente de información procede del portal de datos abiertos del propio Ayuntamiento.

El objetivo es llevar a cabo un análisis específico de los avisos correspondientes al distrito de Moratalaz.

Paso 1: Abrir el proyecto y crear un nuevo Notebook

- Abra el proyecto del ejercicio anterior "Notebook_API.aprx"
- Compruebe de estar conectado al portal correspondiente
- Para crear en Notebook desde ArcGIS Pro vaya a la pestaña de insertar y en la sección de *Project*, haga clic en *New Notebook*.



Para conectar al portal de GIS, primero debemos importar el módulo de gis.

- Añada en la celda vacía la siguiente expresión:

```
In [ ]: from arcgis.gis import GIS
```

- Ejecuta la celda
- Acceda al portal usando argumento en la clase GIS, "Home".

```
[2]: gis=GIS("Home")
     gis
```

```
[2]: GIS @ https://educacionesri.maps.arcgis.com/
```

Paso 2: Buscar la capa Moratalaz Avisos dentro del contenido del portal

- Busque el ítem correspondiente a los avisos recibidos en el Ayuntamiento de Madrid para el distrito de Moratalaz mediante el método `search()`.

Nota: El recurso se denomina "Avisos_Moratalaz". En este ejercicio, para localizar la capa de interés utilizaremos directamente el identificador del ítem

- Busque el ítem de avisos utilizando el siguiente ID: 150f4627aee445ce9fce1db2684a301b y ejecute la celda.

```
dato = gis.content.search("150f4627aee445ce9fce1db2684a301b")
dato

Out[ ]: [<Item title:"Avisos_Moratalaz" type:Feature Layer Collection owner:Tutor_MO>]
```

Puede emplear el módulo IPython para visualizar la lista de resultados obtenidos de una manera más clara y amigable, especialmente si lo combina con un iterador para recorrer los elementos devueltos.

Escriba el siguiente código para poder visualizar el ítem:

```
from IPython.display import display

for item in dato:
    display(item)
```



Dado que se trata de una Feature Layer Collection, el ítem está compuesto por varias Feature Layers. Para identificar qué capas la integran, añadiremos un iterador que nos permita recorrer y visualizar cada una de ellas.

Cree un iterador para extraer los nombres de las capas dentro de Feature Layer Collection:

```
for lyr in item.layers:
    print(lyr.properties.name)
```

```
for lyr in item.layers:
    print(lyr.properties.name)

Moratalaz_Avisos
Barrios
```

La *Feature Layer Collection* de Moratalaz avisos está compuesta por dos *Feature Layers*, **Moratalaz_Avisos** y **Barrios**. La capa Moratalaz_Avisos corresponde a los avisos y la capa Barrios a los límites de los barrios del distrito de Moratalaz.

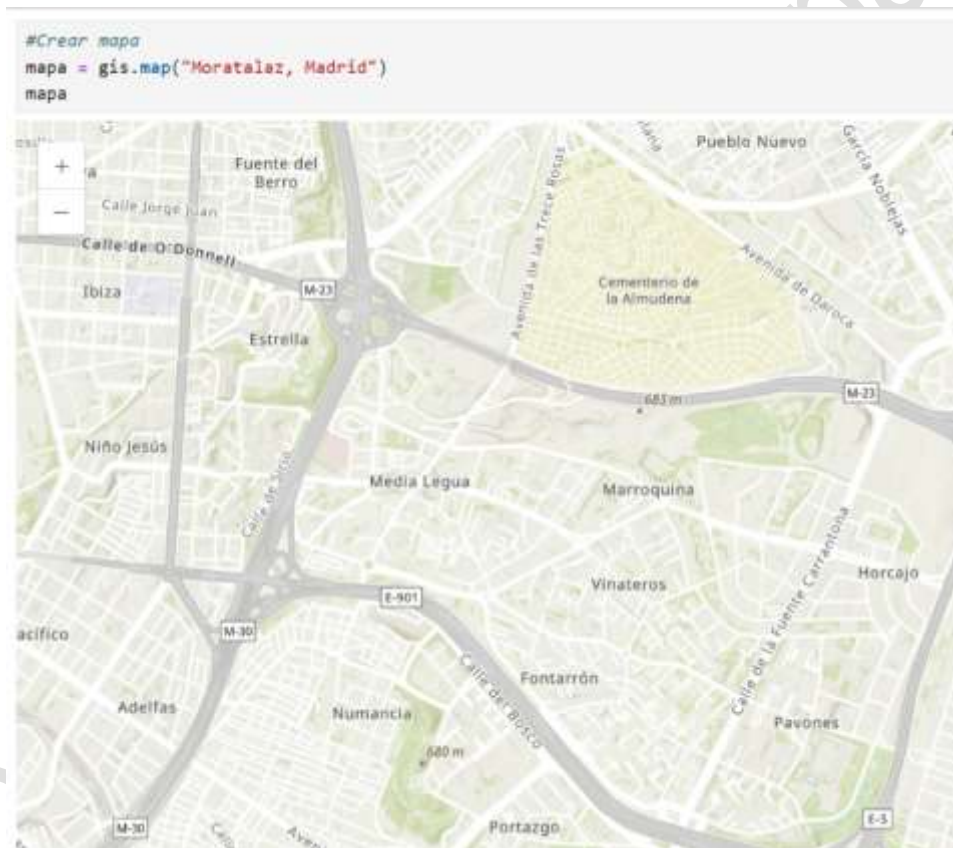
Paso 3: Crear variables que guarden los avisos y los límites de los barrios

- Cree una variable que guarde la capa correspondiente a los avisos, la cual llamará **avisos** y otra que guarde los límites de los barrios, que recibirá el nombre de **barrios**.

```
#Capa de Avisos Recepcionados
avisos = item.layers[0]
#Capa Barrios
barrios = item.layers[1]
```

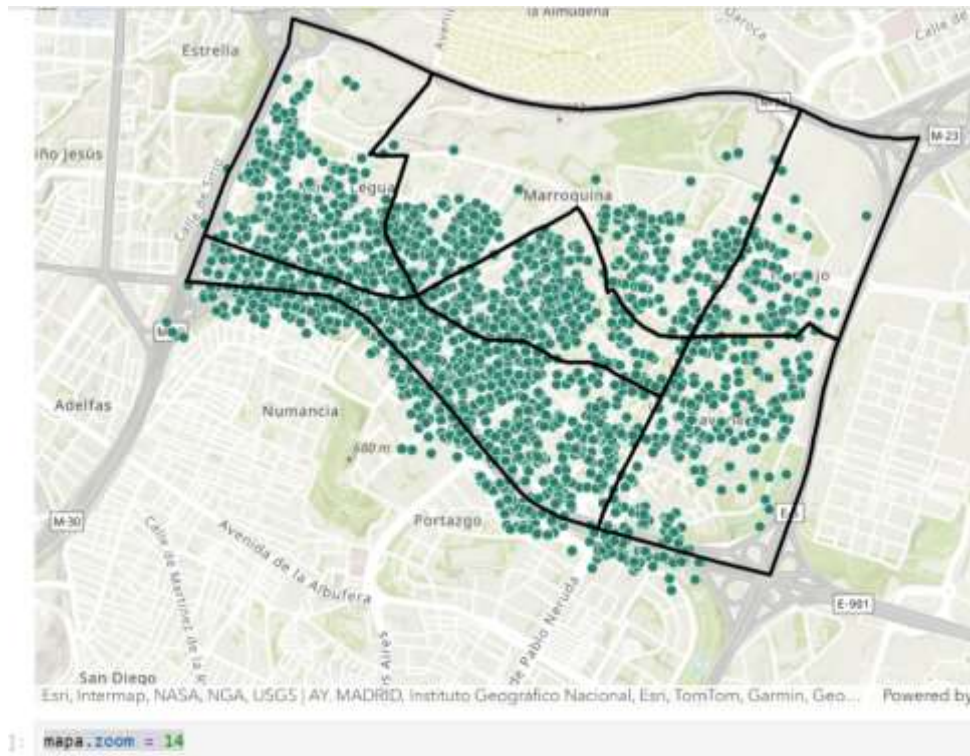
Ahora procederemos a visualizar los **avisos** y **barrios** en un mapa.

- Cree un mapa, centrado en la zona de Moratalaz.



- En la siguiente celda, asigne el valor 14 para el nivel de zoom: `mapa.zoom = 14`
- Añada al mapa creado las capas correspondientes a los **avisos** y **barrios**.

```
mapa.content.add(avisos)
mapa.content.add(barrios)
```



Paso 4: Agregación de los avisos por barrios

Realizará un análisis para identificar cuál de los barrios que integran el distrito de Moratalaz ha recibido un mayor número de avisos. Para ello, utilizará la herramienta `spatial_join()`, utilizando la relación espacial `contains` para contar los puntos contenidos en cada polígono

- En primer lugar, importe el módulo features para poder ejecutar dicha herramienta.

```
from arcgis.features.analysis import aggregate_points
```

- Proceda a configurar la herramienta de agregación de puntos. La capa resultante deberá seguir la siguiente estructura de nomenclatura: **avisos_agregacion_barrios_<iniciales>**.

Nota: Tenga en cuenta que este proceso puede requerir varios minutos para completarse.



Nota: advertencia "InsecureRequestWarning"

Durante la ejecución de herramientas de análisis alojado (por ejemplo, *Aggregate Points*, *Buffer*, *Overlay Layers*, etc.) es posible que aparezca la advertencia siguiente: *InsecureRequestWarning: Unverified HTTPS request is being made to host 'analysis*.arcgis.com'* Esta advertencia procede del módulo Python urllib3 y no indica un error en el análisis ni un problema de seguridad real. ArcGIS Pro gestiona la validación de certificados mediante sus propios componentes internos, por lo que urllib3 no interviene en dicho proceso. Como consecuencia, el módulo interpreta las solicitudes HTTPS como "no verificadas", generando este mensaje de advertencia. La operación puede completarse con normalidad, y el mensaje no afecta a los resultados obtenidos.

Opcionalmente:

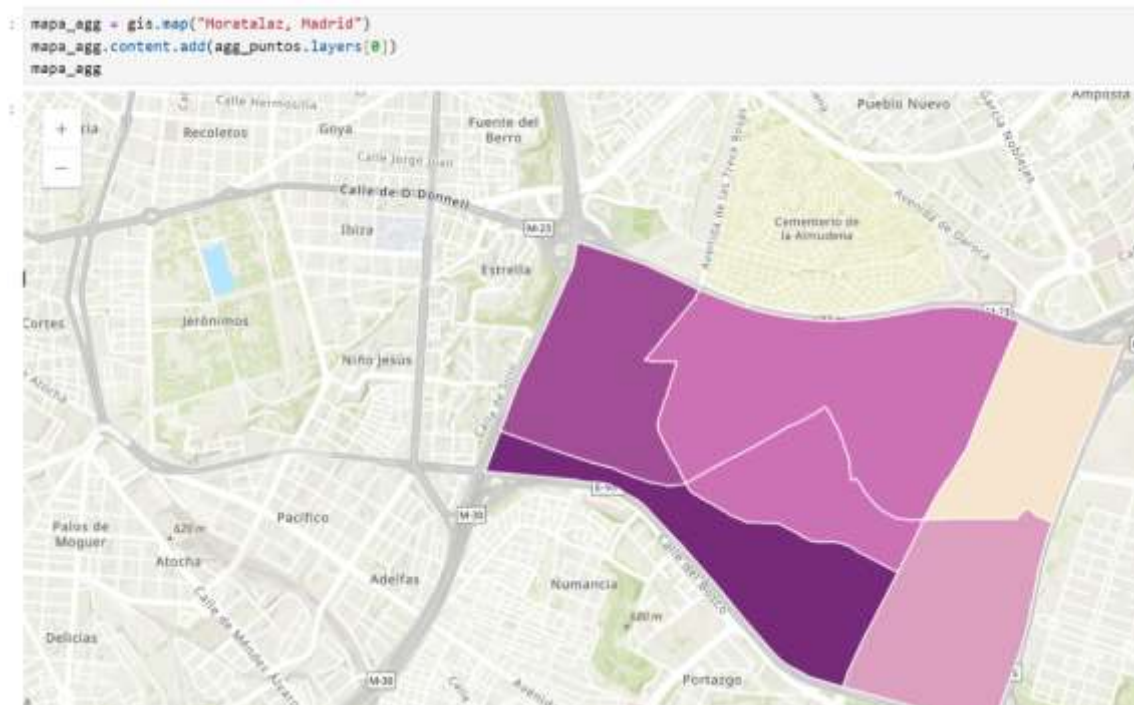
Si se desea suprimir la advertencia para evitar que se repita en la salida del Notebook, puede añadirse la siguiente instrucción al inicio del script:

```
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
```

Si ejecuta nuevamente la herramienta para comprobar su funcionamiento, deberá modificar el nombre de la capa de salida; de lo contrario, se producirá un error indicando que el elemento ya existe en su contenido. En esta segunda ejecución, las advertencias dejarán de aparecer y se mostrará directamente la información relativa al consumo de créditos asociado a la operación.



- Añade el resultado del análisis realizado al mapa web.



Paso 5: Cambiar la simbología del mapa

Para mejorar la visualización del resultado obtenido en el paso anterior y apreciar con claridad las diferencias entre los distintos barrios, modificaremos la simbología aplicada a la capa agregada.

- En primer lugar, identifique el campo que almacena el recuento de avisos por barrio, denominado Point_Count.

```
#Consultar los campos que componen la capa resultado del análisis de agregación de puntos
for field in agg_puntos.layers[0].properties.fields:
    print(f"{field.name}")
```

```
OBJECTID
NOMDIS
CODBAR
CODDISTRIT
CODBARRIO
NOMBRE
Point Count
AnalysisArea
```

- Una vez identificado el campo que contiene el número de avisos por barrio, cree un nuevo mapa con el fin de visualizar de manera más clara cuáles son los barrios que han registrado un mayor número de avisos dentro del distrito.

```
mapa_avisos = gis.map("Moratalaz, Madrid")
mapa_avisos.content.add(agg_puntos.layers[0], options={"opacity": 0.7})

mapa_avisos
```

En la versión actual de la ArcGIS API for Python, la modificación de la simbología se realiza mediante *renderers*. Aunque existen clases como *ClassBreaksRenderer* para definir simbologías por intervalos de forma manual, la forma recomendada y plenamente compatible con el widget de mapa consiste en utilizar el gestor de renderizado (*RendererManager*) junto con las capacidades de *Smart Mapping*. Este enfoque permite generar automáticamente la simbología adecuada sin necesidad de construir manualmente las definiciones de clases.

- Importe la clase *ClassBreakRenderer*.

```
from arcgis.map.renderers import ClassBreaksRenderer
```

- Cree un *renderer* para aplicar la simbología de *Smart Mapping*:

```
renderer_manager = mapa_avisos.content.renderer(0)
smart_mapping_manager = renderer_manager.smart_mapping()
smart_mapping_manager.class_breaks_renderer(
    break_type="color",
    field="Point_Count",
)
```

- 1) **mapa_avisos.content.renderer(0)**: accede al *RendererManager* de la primera capa del mapa (índice 0). Ese objeto es el responsable de leer y modificar la simbología de esa capa.
- 2) **renderer_manager.smart_mapping()**: obtiene un *Smart Mapping manager*, que encapsula las capacidades de *Smart Mapping* de ArcGIS. Es decir, deja que sea ArcGIS quien calcule automáticamente: número de clases, cortes de los intervalos, rampa de color adecuada, etc.


```
# Selección de atributos de interés
df = sdf[['NOMBRE', 'NOMDIS', 'CODDISTRIT', 'Point_Count']]

# Renombrar los atributos
df_new = df.rename(columns={
    "NOMBRE": "Barrio",
    "NOMDIS": "Distrito",
    "CODDISTRIT": "Codigo",
    "Point_Count": "num_Avisos"
})

df_new.head()
```

	Barrio	Distrito	Codigo	num_Avisos
0	Pavones	Moratalaz	14	329
1	Horcajo	Moratalaz	14	109
2	Marroquina	Moratalaz	14	439
3	Media Legua	Moratalaz	14	557
4	Fontarón	Moratalaz	14	843

- Opcionalmente, exporte la información contenida en este nuevo *DataFrame* a un Excel.

```
df_new.to_excel(r"C:\EsriTraining\API\Avisos_Barrios_Moratalaz.xlsx")
```

Nota: Cambie la ruta de salida para que coincida con la ubicación donde quiere que se guarde el Excel que va a exportar.

Paso 7: Visualizar la distribución del número de avisos por barrio en una grafica

Para representar en un histograma los resultados del análisis de la distribución de avisos entre los barrios del distrito de Moratalaz, se utilizará [la librería Matplotlib](#), la cual constituye una de las herramientas fundamentales para la visualización gráfica en Python.

- Importe la librería matplotlib como plt:

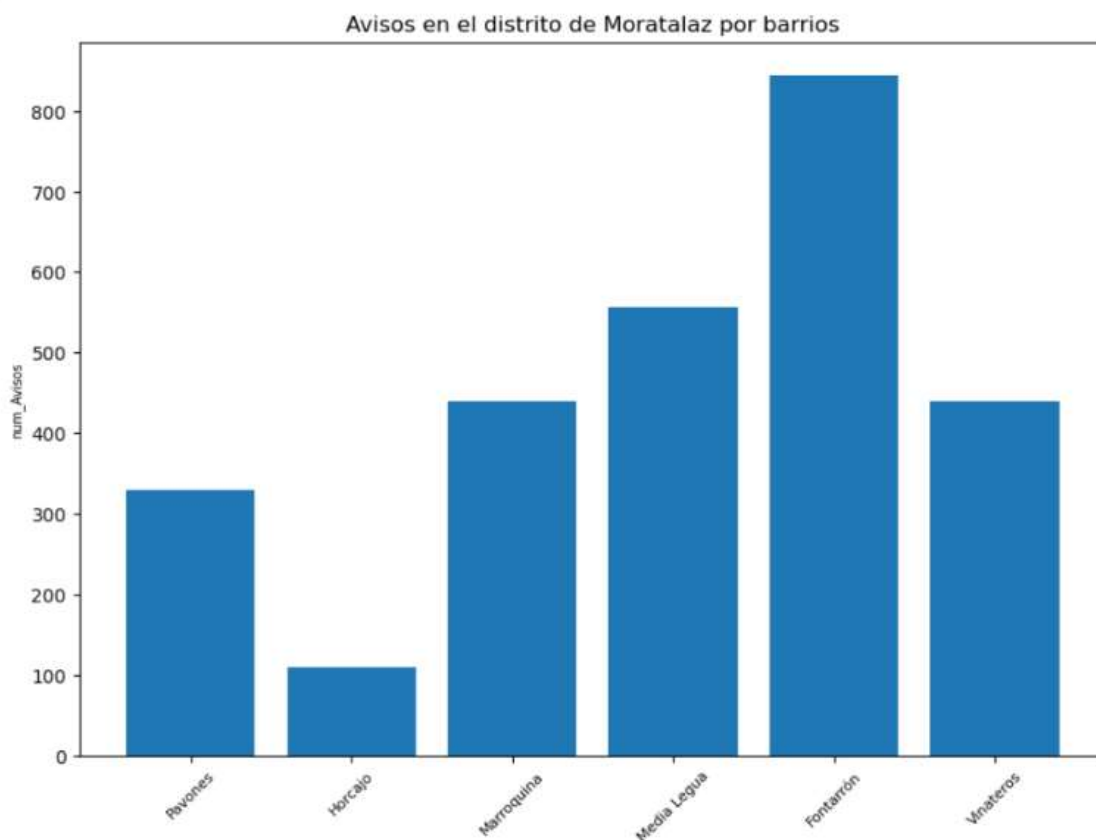
```
import matplotlib.pyplot as plt
```

- Agregue los parámetros ofrecidos para crear un gráfico de barras:

```
%matplotlib inline
plt.figure(figsize=(10,7))
plt.bar(df_new.Barrio, df_new.num_Avisos)
plt.xlabel('Barrio', fontsize=7)
plt.ylabel('num_Avisos', fontsize=7)
plt.xticks(fontsize=8, rotation=45)
plt.title('Avisos en el distrito de Moratalaz por barrios')
plt.show()
```

```
%matplotlib inline
```

```
plt.figure(figsize=(10,7))
plt.bar(df_new.Barrio, df_new.num_Avisos)
plt.xlabel('Barrio', fontsize=7)
plt.ylabel('num_Avisos', fontsize=7)
plt.xticks(fontsize=8, rotation=45)
plt.title('Avisos en el distrito de Moratalaz por barrios')
plt.show()
```



Para guardar en el local en el formato .png se puede utilizar el mismo método **.savefig** que vimos anteriormente con otro ejemplo dentro del entorno Notebooks en ArcGIS Online.

- Agregue una fila antes de visualizar el gráfico con `plt.show()`, sería penúltima fila en el paso anterior:

```
plt.savefig(r"C:\EsriTraining\PAPI\avisos_moratalaz.png", dpi=300,
bbox_inches="tight") #guardado en el local
```

- Compruebe el código final para poder visualizar el gráfico y guardarlo en el local:

```
plt.figure(figsize=(10,7))
plt.bar(df_new.Barrio, df_new.num_Avisos)
plt.xlabel('Barrio', fontsize=7)
plt.ylabel('num_Avisos', fontsize=7)
```

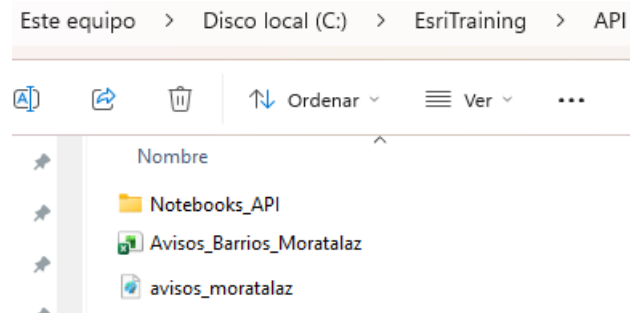
```
plt.xticks(fontsize=8, rotation=45)

plt.title('Avisos en el distrito de Moratalaz por barrios')

plt.savefig(r"C:\EsriTraining\API\avisos_moratalaz.png",      dpi=300,
bbox_inches="tight") #guardado en el local

plt.show()
```

- Compruebe el guardado correcto sobre la ruta de salida: C:\EsriTraining\PAPI\avisos_moratalaz.png en el Explorador de archivos



- Guarde el Notebook como **Ejercicio7_Analisis de avisos.ipynb**
- Guarde el proyecto **Notebook_API.aprx** para no perder los cambios.

6. Introducción al análisis espacial con ArcGIS API for Python

El análisis espacial es uno de los pilares fundamentales de los Sistemas de Información Geográfica. Permite transformar datos geográficos en información útil mediante la aplicación de modelos, reglas espaciales y algoritmos de cálculo sobre entidades vectoriales, redes y datos ráster. ArcGIS API for Python proporciona un entorno flexible y potente para realizar este tipo de análisis desde scripts y notebooks, integrándose directamente con los servicios de ArcGIS Online y ArcGIS Enterprise.

A diferencia de otros entornos puramente locales, el API de Python trabaja bajo un modelo **cliente-servidor**, en el que muchas de las operaciones de análisis se ejecutan como **servicios web GIS**. Esto permite escalar los análisis, reutilizar servicios corporativos y automatizar flujos de trabajo complejos, aunque en determinados casos implica consumo de créditos.

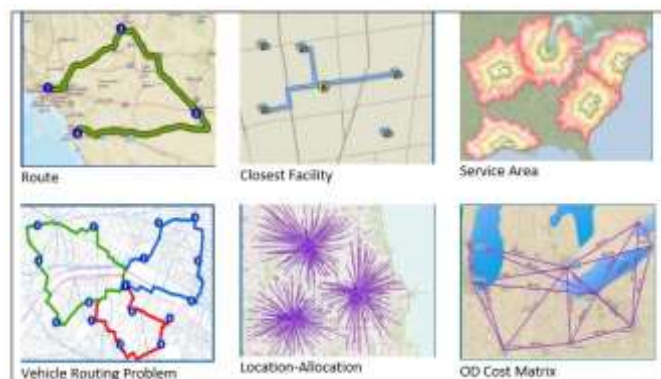
Análisis de redes (Network Analysis)

El análisis de redes permite estudiar la movilidad y la conectividad en un territorio a partir de una red de transporte. Con ArcGIS API for Python es posible acceder a servicios de red para realizar, entre otros, los siguientes análisis:

- Cálculo de rutas óptimas entre puntos.
- Localización de los elementos más cercanos.
- Generación de áreas de servicio en función del tiempo o la distancia.
- Evaluación de accesibilidad a equipamientos o servicios.

Por ejemplo, mediante el módulo **arcgis.network**, se pueden calcular **áreas de servicio** alrededor de centros de salud, colegios o estaciones de emergencia para determinar qué zonas quedan cubiertas en 5, 10 o 15 minutos de desplazamiento. Este tipo de análisis resulta especialmente útil en estudios de planificación urbana, emergencias o logística.

Estos análisis suelen apoyarse en servicios de red alojados en ArcGIS Online o ArcGIS Enterprise, y su ejecución puede implicar consumo de créditos cuando se utilizan servicios online.



Nota: descubra ejemplos reales con la guía de ArcGIS API for Python: [Guide to Network Analysis \(Part 1 - Network Dataset and Network Analysis\) | ArcGIS API for Python | Esri Developer](#)

Análisis de imágenes y ráster (Image & Raster Analytics)

El análisis de imágenes y datos ráster permite trabajar con información continua del territorio, como imágenes satelitales, modelos digitales de elevaciones o productos derivados. ArcGIS API for Python incorpora módulos específicos para este tipo de análisis, facilitando tanto el acceso como el procesamiento de grandes volúmenes de datos.

Entre las tareas más habituales se encuentran:

- Clasificación de imágenes (supervisada y no supervisada).
- Detección de cambios en el territorio.
- Análisis de pendientes, orientaciones y visibilidad.
- Cálculo de índices espectrales (por ejemplo, NDVI).
- Automatización de flujos de trabajo de teledetección.

El módulo **arccgis.raster** contiene clases y funciones de análisis ráster para trabajar con datos ráster y capas de imágenes.

Por ejemplo, es posible analizar imágenes multiespectrales para identificar cambios de uso del suelo a lo largo del tiempo, detectar nuevas construcciones o evaluar el estado de la vegetación.

En entornos avanzados, el API también permite integrar modelos de **aprendizaje automático y deep learning** para la detección automática de objetos o la clasificación de píxeles.

Nota: dirija a la página de la documentación oficial de API para ver como acceder a las capas de imágenes ví API de Python: [Using Imagery Layers | ArcGIS API for Python | Esri Developer](#)



Ventajas del análisis espacial con ArcGIS API for Python

El uso de ArcGIS API for Python para análisis espacial aporta varias ventajas clave:

- Automatización de procesos GIS repetitivos.
- Integración directa con datos y servicios de ArcGIS Online y ArcGIS Enterprise.
- Uso combinado de análisis vectorial, de redes y ráster en un mismo entorno.
- Posibilidad de documentar y reproducir los análisis mediante notebooks.
- Escalabilidad, al apoyarse en servicios web GIS.

En los ejercicios siguientes se profundizará en estos conceptos mediante ejemplos prácticos de análisis de redes y análisis ráster, aplicados a casos reales.

Nota: Este análisis utiliza servicios de red de ArcGIS Online; por tanto, su ejecución consume créditos. Evite repetir el cálculo innecesariamente y reutilice el resultado siempre que sea posible.

Ejercicio 8: Análisis espacial con redes

En este ejercicio usted calculará áreas de servicio alrededor de varios puntos (centros) para identificar zonas cercanas y alejadas según el tiempo de desplazamiento. Para ello utilizará el módulo de redes **arccgis.network** y los servicios de análisis de ArcGIS Online.

Nota: esta operación consume créditos.

Paso 1: Conexión al portal y librerías

- Cree un Nuevo notebook para poder realizar este ejercicio y nómbrelo como **"Ejercicio 8: Análisis espacial con redes"**:
- Importe las librerías necesarias y cree la conexión a su organización:

```
from arccgis.gis import GIS  
gis = GIS("home")  
import pandas as pd
```

Paso 2: Cargar los datos desde el CSV

Usted partirá de un CSV publicado en ArcGIS Online. Va a leerlo con Pandas:

Nota: El URL del CSV: <https://educacionesri.maps.arcgis.com/sharing/rest/content/items/872af856ab34413f8b05b6a59f5bbf43/data>

- Defina la URL del CSV con la variable **centros**:

```
centros =  
pd.read_csv('https://educacionesri.maps.arcgis.com/sharing/rest/content/items/872af856ab34413f8b05b6a59f5bbf43/data')  
centros.head()
```

```
# Partimos de unos datos obtenidos del open data de la Comunidad de Madrid  
centros = pd.read_csv('https://educacionesri.maps.arcgis.com/sharing/rest/content/items/872af856ab34413f8b05b6a59f5bbf43/data')  
centros.head()
```

_OBJETID	PK	NOMBRE	DESCRIPCION-ENTIDAD	HORARIO	EQUIPAMIENTO	TRANSPORTE	DESCRIPCION	ACCESIBILIDAD	CONTENT-URL	COORD
0	1	11867	Centro de Atención a las Adicciones (CAD) de A...	Horario ordinario : De lunes a jueves de 8...	Valoración por equipo interdisciplinar Trata...	Bus: 6 , 8 , 18 , 19 , 22 , 45 , 47 , 76 , 78 ...	Forma de acceso : directa, a través de los ser...	1	http://www.madrid.es/sites/v/index.jsp?vgnextc...	-
1	2	11862	Centro de Atención a las Adicciones (CAD) de L...	Horario ordinario : De lunes a jueves de 8...	Valoración por equipo interdisciplinar Trata...	Bus: 25 , 31 , 138 Metro: Laguna (L 6) , Casa ...	Acceso : directo, a través de los servicios mu...	1	http://www.madrid.es/sites/v/index.jsp?vgnextc...	-
2	3	11863	Centro de Atención a las Adicciones (CAD) de V...	Horario ordinario : De lunes a jueves de 8...	Valoración por equipo interdisciplinar Trata...	Bus: 76 Renfe: Cercanías Puente Alcorcón Bici...	Acceso : Directo a través de los servicios mu...	1	http://www.madrid.es/sites/v/index.jsp?vgnextc...	-
3	4	8880	CMSc Centro Municipal de Salud Comunitaria	Horario de Invierno De lunes a viernes de 9...	NaN	Bus: 34 , 35 Bicicletas: Servicio BiciMadrid	Ámbito de influencia Distrito de Carabanchel	1	http://www.madrid.es/sites/v/index.jsp?vgnextc...	-

Paso 3: Crear un mapa y visualizar los puntos

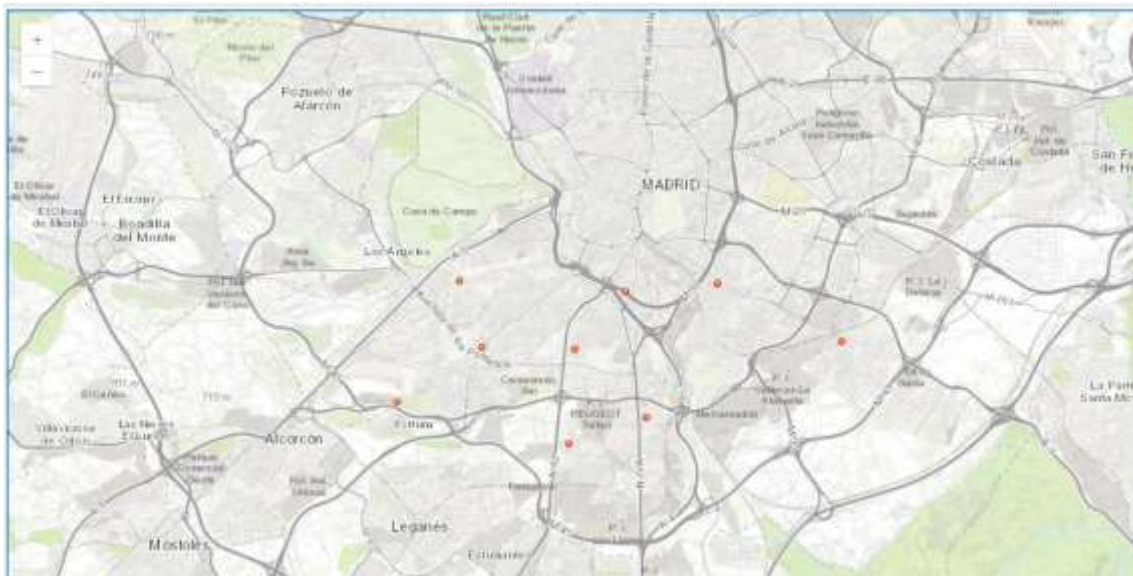
- Cree un mapa centrado en Madrid y muestre el widget:

```
#Crear w1 widget mapa para visualizar los centros de salud
mapa = gis.map("Madrid")
mapa
```

- Añada los puntos al mapa utilizando el método **import_data** que importa un [DataFrame de Pandas](#) (que tiene una columna de direcciones) o un DataFrame espacial de ArcGIS al SIG.

```
#Importar los datos desde csv
featureLayer = gis.content.import_data(centros)
#agregar la capa al widget mapa
mapa.content.add(featureLayer)
```

- Ejecute la celda para poder visualizar los puntos



Paso 4: Trabajar con el módulo `arcgis.network`

- Importe los módulos de red y de entidades (features):

```
# Vamos a utilizar un nuevo módulo del API de Python: network
import arcgis.network as network
import arcgis.features as FeatureLayer
```

Usted utilizará el servicio de áreas de servicio de ArcGIS Online. En ArcGIS Enterprise podría usar servicios propios, pero en este ejercicio se utilizará Online.

- Obtenga el URL del servicio de red a través de la siguiente sintaxis

```
service_area_url = gis.properties.helperServices.serviceArea.url
service_area_url
```

```
# Nuestro "GIS" contiene un servicio de red disponible listo para usar por nosotros (algunas herramientas consumen créditos)
# Este servicio es configurable y podemos usar el nuestro propio en ArcGIS Enterprise aunque aquí estamos usando el servicio de AGOL

service_area_url = gis.properties.helperServices.serviceArea.url
service_area_url
```

```
'https://route.arcgis.com/arcgis/rest/services/World/ServiceAreas/NAService/ServiceArea_World'
```

- Cree el objeto `ServiceAreaLayer` con la siguiente sintaxis:

```
sa_layer = network.ServiceAreaLayer(service_area_url, gis=gis)
sa_layer
```

```
# Obtenemos el servicio concreto para realizar operaciones de tipo service área
sa_layer = network.ServiceAreaLayer(service_area_url, gis=gis)
sa_layer
```

```
<ServiceAreaLayer url:'https://route.arcgis.com/arcgis/rest/services/World/ServiceAreas/NAService/ServiceArea_World'>
```

Vamos a utilizar, como se ve en el mapa superior, un Service Area. Este tipo de servicio nos permite calcular la distancia que podemos recorrer en cierto tiempo alejándonos de un punto concreto del mapa.

En nuestro caso, vamos a calcular lo que podemos alejarnos de los colegios de Madrid en 1, 3 y 5 minutos. De este modo, las zonas que estén más lejos de los colegios serán a las que tardemos en llegar al menos 5 minutos. Consideraremos cercanas aquellas zonas que estén a 1-3 minutos de distancia.

Paso 5: Ejecutar el cálculo de áreas de servicio (1, 3 y 5 minutos)

En este pasó usted calculará polígonos de área de servicio para 1, 3 y 5 minutos.

Vamos a aplicar esta funcionalidad a todos nuestros centros de salud.

- Aplique el método `.query()` simulando un 1=1 para luego calcular 3 saltos drive-time.

```
centros_salud = featureLayer.query()
```

Usamos todos nuestros centros de salud y calculamos 3 saltos drive-time, de este modo obtenemos las distancias andando a 1 minuto, 3 minutos y 5 minutos de cada punto

- Ejecute el análisis con la siguiente sintaxis:

```
result = sa_layer.solve_service_area(featureLayer.query(),
default_breaks=[1,3,5])
result.keys()
```

El resultado suele incluir un conjunto de mensajes e información de salida. La parte que nos interesa es la colección de polígonos (normalmente saPolygons o un nombre equivalente).

```
# Usamos todos nuestros centros de salud y calculamos 3 saltos drive-time, de este modo obtenemos
# las distancias andando a 1 minuto, 3 minutos y 5 minutos de cada punto
result = sa_layer.solve_service_area(featureLayer.query(), default_breaks=[1,3,5])
```

```
# Revisamos el resultado porque es un objeto complejo compuesto de varios "sub-objetos"
result.keys()
```

```
dict_keys(['requestID', 'saPolygons', 'messages'])
```

Conviene revisar muy bien el resultado de esta herramienta de geoprocésamiento. El resultado contiene una serie de diccionarios que contienen toda la información que necesitamos, pero no es fácil gestionarla.

Vamos a ver cómo el resultado es un diccionario con mensajes que nos envía ArcGIS Online y con una batería de polígonos llamados "saPolygons". Estos polígonos son las formas verdes, amarillas y rojas que indican la distancia o proximidad a los centros de salud.

- Obtenga los polígonos:

```
result['saPolygons'].keys()
```

```
# Vemos que hay polígonos que contienen features dentro de "saPolygons"
result['saPolygons'].keys()
```

```
dict_keys(['fieldAliases', 'geometryType', 'spatialReference', 'fields', 'features'])
```

Paso 6: Preparar el resultado para representarlo en el mapa

Del mismo modo que hicimos antes, para poder mostrar elementos en un mapa, necesitamos contar con las geometrías y con la simbología necesaria para aplicar a dichas geometrías.

Vamos a comenzar creando las Features con las geometrías existentes en el resultado y también los atributos. En este caso, vamos a contar con los atributos ya que se incluye el atributo "ToBreak" que representa de cada uno de los polígonos el nivel de distancia que tienen (1, 3 o 5 minutos):

- Importe las clases Feature y FeatureSet dentro del módulo arcgis.features

```
from arcgis.features import Feature, FeatureSet
```

- Cree una lista con los polígonos que incluye el resultado del service área

```
listaPoligonos = []
```

- Para cada feature que viene en el resultado conviértala en un objeto Feature del API y guarde tanto su geometría como sus atributos

```
for polygon_dict in result['saPolygons']['features']:
    feature = Feature(polygon_dict['geometry'],
polygon_dict['attributes'])
    listaPoligonos.append(feature)
```

Ya que teníamos 9 puntos y calculamos 3 drive-times para cada uno, el resultado serán 27 polígonos (9 representan 1 minuto de cercanía, 9 representan 3 minutos y 9 representan 5 minutos)

- Calcule la lista de polígonos para comprobar la cantidad

```
len(listaPoligonos)
```

Si revisamos los atributos de los polígonos, podemos ver que los atributos "FromBreak" y "ToBreak" marcan para cada polígono en qué tramo se encuentran (1, 3 ó 5)

- Revise los atributos con la siguiente sintaxis:

```
listaPoligonos[0].attributes
```

```

: # Si revisamos los atributos de los polígonos,
  listaPoligonos[0].attributes

: {'ObjectID': 1,
  'FacilityID': 9,
  'Name': 'Location 9 : 3 - 5',
  'FromBreak': 3,
  'ToBreak': 5,
  'Shape_Length': 0.8964469824531499,
  'Shape_Area': 0.0011768164560380684}

```

Paso 7: Definir la simbología según el tiempo (1, 3, 5)

Usted aplicará una simbología simple (el más próximo a los centros de salud será verde y el más alejado rojo)

- **1 minuto:** verde
- **3 minutos:** amarillo
- **5 minutos:** rojo

➤ Defina los símbolos:

```

colors = {5: [255, 0, 0, 90],
          3: [255, 255, 0, 90],
          1: [0, 128, 0, 90]}

fill_symbol = {"type": "esriSFS", "style": "esriSFSSolid", "color":
[255,255,255,255]}

```

Vamos a dejar un objeto "fill_symbol" con un color por defecto. Posteriormente, a iremos modificando este objeto en función del valor del atributo de cada polígono:

➤ Importe las clases necesarios del módulo `arcgis.map.symbols`

```

from arcgis.map.symbols import SimpleLineSymbolEsriSLS,
SimpleFillSymbolEsriSFS

```

➤ Cree una variable `poly` de la lista de Polígonos obtenidos:

```

poly = listaPoligonos[0]

poly

```

```
poly = listaPoligonos[0]
poly

{"geometry": {"rings": [[[-3.832285203999959, 40.39493534700006], [-3.8293656919999535, 40.39224041400007], [-3.8271199139999794, 40.3199996, 40.38482934600006], [-3.815891824999928, 40.38438019100005], [-3.8116240469999525, 40.38288814600006], [-3.8084799569999745, 40.9999605, 40.37629539000005], [-3.806907912999975, 40.37539707900004], [-3.806740227999967, 40.37472634000005], [-3.8082553799999346, 40.9359999395, 40.38123618200005], [-3.816115601999968, 40.382358991000046], [-3.8219546249999325, 40.38288814600006], [-3.82667075899996, 383514999733, 40.39483703600004], [-3.832285203999959, 40.39493534700006], [-3.8260599069999785, 40.330481528000035], [-3.727182795, [-3.729283995999967, 40.33285356500008], [-3.7290777299999415, 40.33317645400007], [-3.7298777299999415, 40.336320543000056], [-3.728774], [-3.7298777299999415, 40.34283329900006], [-3.742804930999994, 40.35069352200003], [-3.744475285999954, 40.35136725500007], [-3.75005], [-3.7487422639999295, 40.34777401000008], [-3.7476193749999425, 40.348447744000055], [-3.74717021999993, 40.34799858800005], [-3.7400066], [-3.7476193749999425, 40.344629921000035], [-3.7491914199999314, 40.34597738800005], [-3.754581286999951, 40.34597738800005], 15956680008], [-3.7566824869999524, 40.34036294300006], [-3.759072842999956, 40.34268872100003], [-3.758623686999954, 40.3435070320000459773880005], [-3.760195731999943, 40.34575281000008], [-3.7615431989999593, 40.34440534400005], [-3.7628906649999294, 40.345079077040.34440534400005], [-3.76895426599998, 40.343057877000035], [-3.76895426599998, 40.34193498800005], [-3.769627990999953, 40.3408120913, 40.337892587000056], [-3.7716491989999668, 40.33879089900006], [-3.7716491989999668, 40.341036677000034], [-3.770975465999925, 40.34999557, 40.34350703200005], [-3.775017065999928, 40.34373161000008], [-3.7770390659999293, 40.34260872100003], [-3.779060266999977, 40.6889999477, 40.344854499000064], [-3.7813060439999617, 40.34440534400005], [-3.7822043559999656, 40.345303655000066], [-3.78197977799993.7797399999962, 40.346651121000036], [-3.776353329999943, 40.34710027700004], [-3.7759161769999423, 40.34754943300004], [-3.7714245], [-3.773894976999941, 40.34777401000008], [-3.7747932879999553, 40.34867232200003], [-3.773894976999941, 40.34979521100007], [-3.7700007], [-3.7725475099999244, 40.35226556600003], [-3.778611110999975, 40.350468944000056], [-3.782653510999978, 40.348447744000055], 36600003], [-3.7819797779999362, 40.35114267700004], [-3.773894976999941, 40.35428676600003], [-3.773424621999927, 40.35091610000006], 40989000045], [-3.7718737769999393, 40.35451134400006], [-3.7707508879999523, 40.35608330900005], [-3.7707508879999523, 40.3572062780040.35720627800005], [-3.7642081899999777, 40.35610704700008], [-3.7630866409999704, 40.35608330900005], [-3.763153420999959, 40.356117631153420999959]
```

Paso 8: Añadir las áreas de servicio al mapa con su simbología

Disponemos de un conjunto de polígonos (áreas de servicio) devueltos por el análisis de red. Cada polígono incluye:

- **Geometría** (sus anillos o *rings*).
- **Atributos**, entre ellos ToBreak, que indica el umbral temporal asociado (por ejemplo, 1, 3 o 5 minutos).

El objetivo es **representar esos polígonos en el mapa** aplicando un color distinto según el valor de ToBreak.

- Importe un objeto **Polygon** con estructura GIS formal, listo para ser consumido por operaciones de visualización y geoprocésamiento desde el módulo `arcgis.geometry`

```
from arcgis.geometry import Polygon
```

- Construya la geometría a partir de los anillos (*rings*):

```
geom = Polygon({
    "rings": poly.geometry['rings'],
    "spatialReference": {"wkid": 4326}
})
type(geom)
```

Con la cual se extrae la geometría original desde un objeto `poly` (un polígono del resultado), recuperando sus anillos (*rings*), que son las coordenadas que definen el contorno del polígono y definiendo explícitamente el sistema de referencia espacial con `wkid: 4326` (WGS84).

Nota: Si no se especifica, el mapa puede interpretar mal las coordenadas, o representarlas en una ubicación incorrecta.

`Type(geom)` se utiliza únicamente para verificar que `geom` es realmente un objeto del tipo `arcgis.geometry.Polygon`. Es un paso de validación útil durante el aprendizaje.

- Ejecuta la celda para comprobar el tipo de geometría obtenida:

```
geom = Polygon({
    "rings": poly.geometry['rings'],
    "spatialReference": {"wkid": 4326}
})
type(geom)

arcgis.geometry._types.Polygon
```

- Para cada polígono resultado de nuestro FeatureSet visualízelo según corresponda (1, 3 ó 5)

```
for poligono in listaPoligonos:

    geom = Polygon({
        "rings": poligono.geometry['rings'],
        "spatialReference": {"wkid": 4326}
    })

    color = colors.get(poligono.attributes['ToBreak'], [255, 255, 255,
255]) # Color neutro si no se encuentra

    fill_symbol = SimpleFillSymbolEsriSFS(
        **{
            "type": "esriSFS",
            "style": "esriSFSSolid",
            "color": color,
            "outline": {"color": [153, 153, 153, 255], "width":
0.375},
        })

    mapa.content.draw(geom, symbol=fill_symbol)

mapa
```

Primero se hace una iteración sobre la lista de polígonos del análisis. Cada iteración procesa **un polígono** y lo representa con la simbología adecuada.

Luego se repite el proceso de construcción de la geometría para cada polígono:

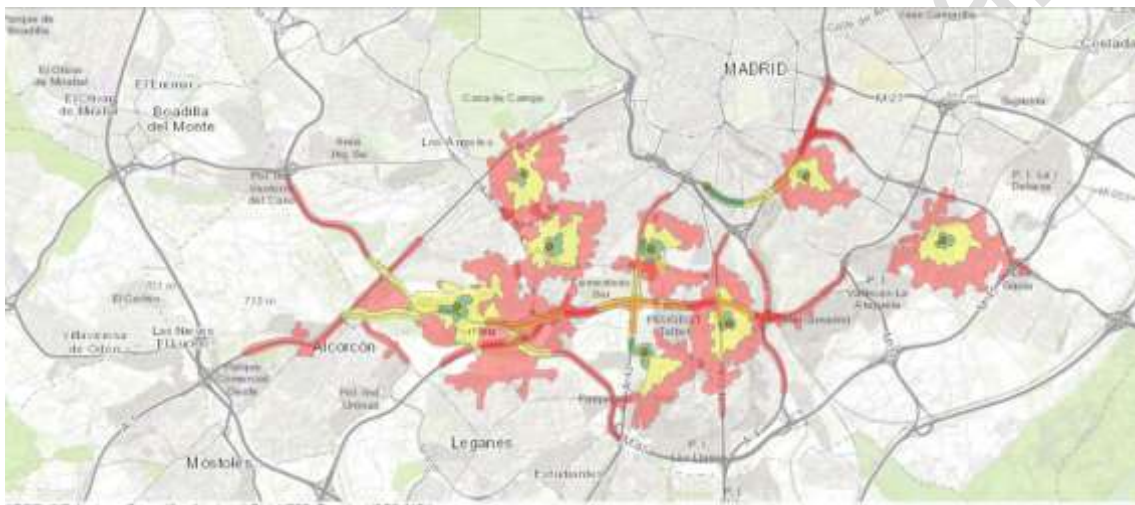
- Se garantiza que cada elemento se representa con geometría formal de ArcGIS.

- Se fuerza el sistema de referencia para evitar problemas de reproyección o interpretación.

Con la variable color se selecciona un color mediante el atributo **ToBreak**. Esto permite controlar la simbología de forma limpia y evita errores si aparece algún valor inesperado.

Y por último con la variable **fill_symbol** contruimos el símbolo de relleno. La clase/constructor **SimpleFillSymbolEsriSFS** genera un símbolo compatible con ArcGIS (habitualmente del entorno de simbología o renderers). Su finalidad es encapsular la definición del símbolo para poder aplicarlo directamente al mapa.

La sintaxis **mapa.content.draw()** permite representar el resultado en el mapa. Este método es útil cuando se desea representar geometrías rápidamente sin publicar capas ni crear servicios en el portal.



- Guarde el Notebook para no perder los cambios.

Interpretación del resultado

- Las zonas cubiertas por el polígono de **1 minuto** representan las áreas más próximas.
- Las zonas cubiertas por **3 minutos** representan proximidad media.
- Las zonas fuera de **5 minutos** pueden considerarse zonas con menor accesibilidad desde esos puntos.

Ejercicio 9: Análisis de imágenes y análisis ráster

En este ejercicio se aprenderá a acceder a imágenes satelitales, visualizarlas en un mapa interactivo y aplicar funciones ráster al vuelo utilizando ArcGIS API de Python. Además, se mostrará cómo exportar el resultado de estas visualizaciones.

Uso de contenidos del Living Atlas

Además de trabajar con datos propios, ArcGIS pone a disposición de los usuarios una gran cantidad de contenidos listos para su uso a través del **Living Atlas of the World**. El Living Atlas es una colección curada de datos geográficos oficiales y de alta calidad, accesibles directamente desde ArcGIS Online y ArcGIS Enterprise.

Entre estos contenidos se incluyen, entre otros:

- Imágenes satelitales de **Landsat**
- Imágenes satelitales de **Sentinel**
- Modelos digitales del terreno
- Capas temáticas globales

Estos datos pueden consumirse directamente desde un Notebook, sin necesidad de descargarlos previamente.

Paso 1: Acceder a las imágenes directamente con ArcGIS Living Atlas

- Cree un Nuevo notebook para poder realizar este ejercicio y nómbrelo como **"Ejercicio 9: Image Layer y Raster Analytics"**:
- Importa el módulo ráster para poder trabajar con imágenes

```
from arcgis.raster import analytics
analytics.is_supported(gis)
```

- El módulo `arcgis.raster.analytics` proporciona acceso a capacidades de análisis ráster.
- El método `is_supported(gis)` comprueba si su organización (ArcGIS Online o Enterprise) tiene habilitado el análisis ráster.
- Este paso es importante para evitar errores antes de ejecutar operaciones más avanzadas.

Importar el módulo raster

Para poder trabajar con imágenes

```
from arcgis.raster import analytics
analytics.is_supported(gis)
```

True

- Acceda a datos Landsat del Living Atlas

```
landsat_item = gis.content.search('title: Multispectral Landsat',  
'Imagery Layer', outside_org=True)[0]  
landsat_item
```

- Se realiza una búsqueda en el portal mediante `gis.content.search()`.
- Se buscan ítems de tipo **Imagery Layer** cuyo título contenga *Multispectral Landsat*.
- El parámetro `outside_org=True` permite acceder a contenidos públicos del Living Atlas.
- El resultado es un **Item**, que puede contener una o varias capas de imágenes.

Acceso a las imágenes directamente con ArcGIS Living Atlas

Landsat



- Cree el widget mapa y visualice la capa de Landsat

```
landsatMap = gis.map('Lodosa')  
landsatMap  
landsatMap.content.add(landsat_item.layers[0])
```

- Habilite el control temporal del mapa con el método `.time_slider.enabled = True`

```
landsatMap.time_slider.enabled = True
```

- Ejecuta las celdas para obtener esta visualización:



Paso 2: Acceso a datos Sentinel del Living Atlas

De forma análoga a Landsat, se buscan capas Sentinel. Se selecciona el primer resultado, que corresponde a un servicio de imágenes Sentinel.

- Busque las imágenes Sentinel de la misma forma con el método `search()`:

```
sentinel_items = gis.content.search(
    'title: Sentinel',
    'Imagery Layer',
    outside_org=True
)
sentinel_item = sentinel_items[0]
sentinel_item
```

Sentinel

```
sentinel_items = gis.content.search('title: Sentinel', 'Imagery Layer', outside_org=True)
sentinel_item = sentinel_items[0]
sentinel_item
```



Sentinel-2 Views
Global Sentinel-2 Level-1C multispectral imagery. This Imagery Layer is sourced from Esri.
Imagery Layer by esri
Last Modified: September 02, 2025
39 comments, 7707571 views

- Visualice Sentinel en un nuevo mapa con la variable `m`:

```
m = gis.map('Lodosa')
m
```

```
sentinel = sentinel_item.layers[0]  
sentinel  
m.content.add(sentinel_item.layers[0])
```

- Ejecuta las celdas para obtener esta visualización:



Paso 3: Trabajar con funciones ráster

Cada ImageLayer dispone de un conjunto de funciones ráster. Estas funciones se aplican directamente sobre los píxeles del servicio. No generan nuevos datasets ni modifican la imagen original. El bucle for permite conocer qué funciones están disponibles para la capa Sentinel.

- Ejecute la celda con el siguiente código:

```
for rasterfunc in sentinel.properties.rasterFunctionInfos:  
    print(rasterfunc.name)
```

El resultado de ejecución es la lista de todas las funciones ráster disponibles:

```
# Vamos a ver las funciones raster disponibles. Estas funciones se a
for rasterfunc in sentinel.properties.rasterFunctionInfos:
    print(rasterfunc.name)

Natural Color with DRA
Agriculture with DRA
Bathymetric with DRA
Color Infrared with DRA
Short-wave Infrared with DRA
Geology with DRA
Natural Color
Agriculture
Bathymetric
Geology
Short-wave Infrared
Color Infrared
NDVI Colormap
NDMI Colorized
Normalized Burn Ratio
Normalized Difference Built-Up Index (NDBI)
NDVI Raw
NDVI - VRE only Raw
NDVI - with VRE Raw
NDWI Raw
NDWI - with VRE Raw
None
```

- Importe la función apply, utilizada para aplicar funciones ráster dinámicamente.

```
from arcgis.raster.functions import apply
```

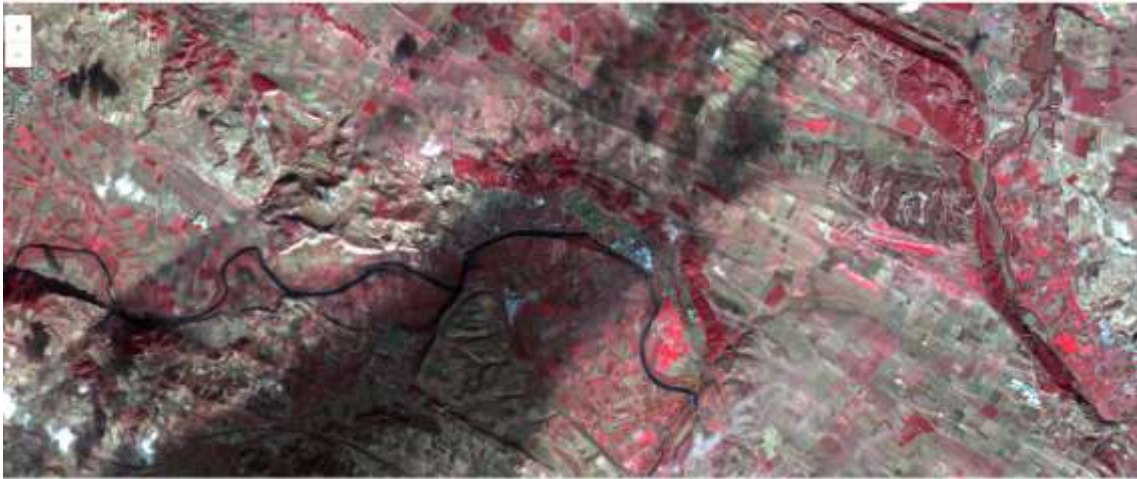
- Aplique la función raster que corresponda a la visualización en el color infrarrojo:

```
# Vamos a aplicar algunas funciones raster para ver el resultado
inf = apply(sentinel, 'Color Infrared with DRA')
m.content.add(inf)
```

Aplicando la función ráster para poder visualizar en el color infrarrojo

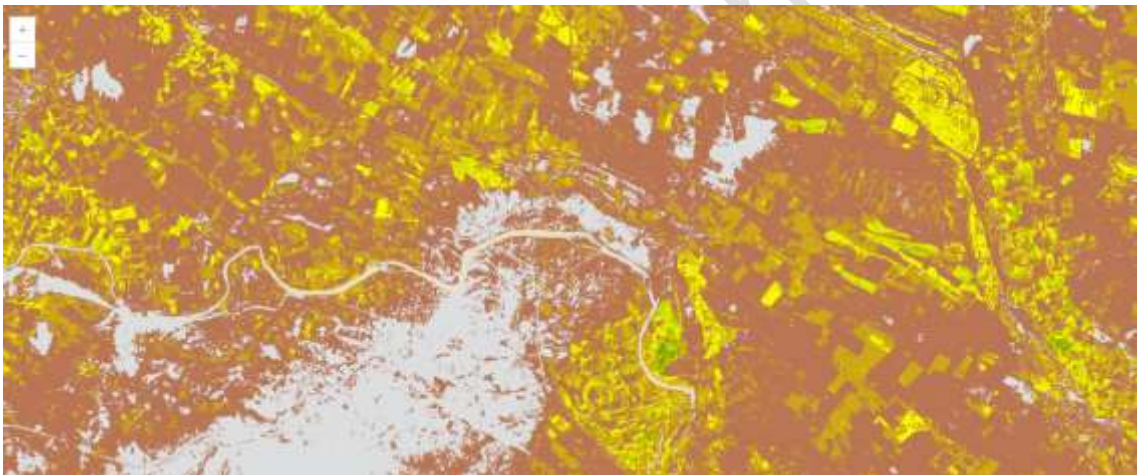
```
# Vamos a aplicar algunas funciones raster para ver el resultado
inf = apply(sentinel, 'Color Infrared with DRA')
m.content.add(inf)
```

El resultado en el widget mapa de Sentinel aparece así:



- Aplique la función raster que corresponda a la visualización del cálculo del índice NDVI:

```
ndvi = apply(sentinel, 'NDVI Colormap')  
m.content.add(ndvi)
```



Paso 4: Guardar imágenes

Primero se obtiene la extensión actual del mapa que se utilizará para exportar la imagen visible.

- Ejecuta la celda con el siguiente código: `m.extent`

```
m.extent
```

```
{'spatialReference': {'latestWkid': 3857, 'wkid': 102100},  
  'xmin': -245503.25013191157,  
  'ymin': 5218239.5070153,  
  'xmax': -215138.64064566625,  
  'ymax': 5229705.061258062}
```

- Cree una variable `img` para exportar una imagen con el método `.export_image()`:

```
img = sentinel.export_image(bbox=m.extent, size=[900, 450], f='image',
export_format='jpg')

img
```

El método **export_image()** genera una imagen estática a partir del servicio ráster. Los parámetros clave son:

- o bbox: área a exportar.
- o size: tamaño de la imagen en píxeles.
- o export_format: formato de salida (JPG en este caso).

El resultado es una imagen en memoria.

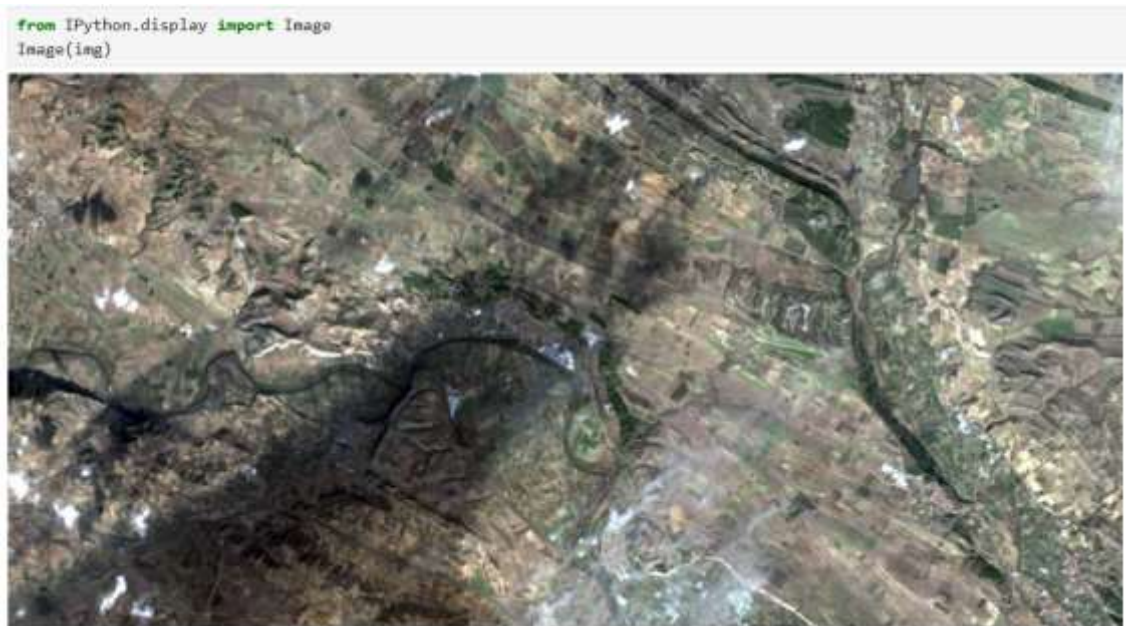
```
img = sentinel.export_image(bbox=m.extent, size=[900, 450], f='image', export_format='jpg')
img
```

```
b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x00\x01\x00\x00\xff\xdb\x00C\x00\x08\x01
\x1a\x1c\x1c$.\'",#\x1c\x1c(7),01444\x1f\'9=82<.342\xff\xdb\x00C\x01\t\t\t\x0c\x0b\x0c\x18\
1"\x00\x02\x11\x01\x03\x11\x01\xff\xc4\x00\x1f\x00\x00\x01\x05\x01\x01\x01\x01\x01\x00\x0
\x04\x03\x05\x05\x04\x04\x00\x00\x01}\x01\x02\x03\x00\x04\x11\x05\x12!1A\x06\x13Qa\x07"q\x142
tuvxyz\x83\x84\x85\x86\x87\x88\x89\x8a\x92\x93\x94\x95\x96\x97\x98\x99\x9a\xa2\xa3\xa4\xa5\x
\xd7\xd8\xd9\xda\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xef\x1f2\x1f3\x1f4\x1f5\x1f6\x1f7\x1f8\x1f9\
5\x06\x07\x08\t\n\x0b\xff\xc4\x00\xb5\x11\x00\x02\x01\x02\x04\x04\x03\x04\x07\x05\x04\x04\x00
\n\x16$4\xe1%\xf1\x17\x18\x19\x1a&\'()*56789: CDEFGHIJSTUVWXYZcdefghijstuvwxyz\x82\x83\x84\x85
b6\xb7\xb8\xb9\xba\xbc\xcd\xce\xcf\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e
\x11\x00?\x00\xe3\xb5\x1b[\x8dGZ\x9eh\x1d\x99\x14\xe0\x10:\x01Q\x0c9}\xac\xea\x84\xb6\xfaJ\x8
\x88$^x\xee?1\x1f9V\x0eVW-S|\xfe\xa75\xa7\xe8\x1a\x0c4hZ\x1f2R\x92\xb1\xce\x0f?\xca\xb4\x7f\xb3\
dA\x14\xa7\xee\xb3\xb61P\xaa\xb6m\xec\x92F"k\x92[0\x176\x1f3\xec=\x1d\x90\xaf\xff\x00\xae\xae\
dd\x1b\x03\xcb\x8b\x15\xaa\xd4\xce\x05\x9d0G\x06\xef\xa1\x10\x0c3j\x04ny\x92F\x03\xe82j\x9d\x
\xe4\x92\x7fJ\xa2\x12\x06\xe4\x91\xdcy2\xa8V\x1f7\x0f\xb4dT\x82]\xf7$\xaa\x9d\x8a2r1Ue\x95-\x
```

- Para poder visualizar la imagen directamente en el Notebook, ejecute la siguiente sintaxis:

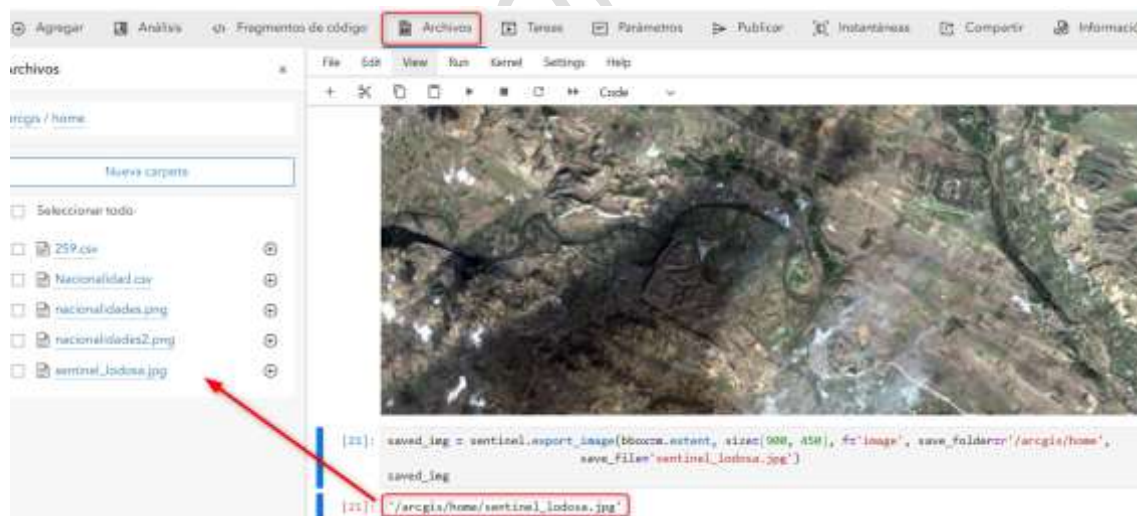
```
from IPython.display import Image

Image(img)
```



Para guardar la imagen dentro del entorno físico de Notebook, ejecute lo siguiente:

```
saved_img = sentinel.export_image(bbox=m.extent, size=[900, 450],
f='image', save_folder=r'/arcgis/home',
save_file='sentinel_lodosa.jpg')
saved_img
```



save_folder indica la carpeta de destino, save_file define el nombre del archivo.

Nota: Este paso permite conservar el resultado para informes o documentación.

- Guarde el Notebook para no perder los cambios.

Conclusión

Con este ejercicio usted ha aprendido a:

- Acceder a imágenes satelitales del Living Atlas.
- Visualizarlas en un mapa interactivo con control temporal.

- Aplicar funciones ráster dinámicamente sin generar nuevos datasets.
- Exportar el resultado visual como imagen.

Este flujo de trabajo es especialmente útil para análisis exploratorios, estudios ambientales y demostraciones rápidas de capacidades ráster dentro del ecosistema ArcGIS.