Programación para *Data Science*

Unidad 4: Librerías científicas en Python

En este Notebook encontraréis un conjunto de **actividades evaluables** como PEC de la asignatura.

Veréis que cada una de ellas tiene asociada una puntuación que indica el peso que tiene la actividad sobre la nota de la PEC. Adicionalmente, hay un ejercicio opcional, que no tiene puntuación dentro de la PEC, pero que se valora al final del semestre de cara a conceder las matrículas de honor y redondear las notas finales. Podéis sacar la máxima nota de la PEC sin necesidad de hacer este ejercicio! El objetivo de este ejercicio es que sirva como pequeño reto para los estudiantes que quieran profundizar en el contenido de la asignatura.

Además, veréis que todas las actividades tienen una etiqueta que indica los recursos necesarios para llevarla a cabo. Hay tres posibles etiquetas:

- `NM` **Sólo materiales**: las herramientas necesarias para realizar la actividad se pueden encontrar en los materiales de la asignatura.
- `EG` **Consulta externa guiada**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, pero el enunciado contiene indicaciones de dónde o cómo encontrar la información adicional necesaria para resolver la actividad.
- `EI` **Consulta externa independente**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, y el enunciado puede no incluir la descripción de dónde o cómo encontrar esta información adicional. Será necesario que el estudiante busque esta información utilizando los recursos que se han explicado en la asignatura.

Es importante notar que estas etiquetas no indican el nivel de dificultad del ejercicio, sino únicamente la necesidad de consulta de documentación externa para su resolución. Además, recordad que las **etiquetas son informativas**, pero podréis consultar referencias externas en cualquier momento (aunque no se indique explícitamente) o puede ser que podáis hacer una actividad sin consultar ningún tipo de documentación. Por ejemplo, para resolver una actividad que sólo requiera los materiales de la asignatura, puedéis consultar referencias externas si queréis, ya sea tanto para ayudaros en la resolución como para ampliar el conocimiento!

En cuanto a la consulta de documentación externa en la resolución de los ejercicios, recordad **citar siempre la bibliografía utilizada** para resolver cada actividad.

Ejercicios para la PEC

A continuación, encontraréis los **ejercicios que debéis completar en esta PEC** y que forman parte de la evaluación de esta unidad.

NumPy - Ejercicios

Ejercicio 1

Crea y muestra por pantalla las matrices siguientes:

- Una matriz A 3x3 que tenga valores del 1 al 9 ordenados por fila.
- Una matriz B 3x3 de números enteros aleatorios (*random*)
- Una matriz C 3x2 que tenga valores de 1 en la 1ª columna y valores de 5 en la 2ª columna.

Considerando las matrices del apartado anterior, muestra por pantalla el resultado de las siguientes operaciones:

- Sumar las matrices A y B
- Multiplicar las matrices A y C. Explica brevemente qué método de multiplicación has elegido y el resultado obtenido.

(0.5 puntos) `NM`

In [1]:

```
# Respuesta
```

Ejercicio 2

Crea una matriz 10 x 10 y rellenala con muestras aleatorias a partir de una distribución normal estándar. Comprueba que la media y la desviación estándar son bastante cercanas a 0 y 1 respectivamente.

(0.5 puntos) NM

In [2]:

```
# Respuesta
```

Ejercicio 3

Crea una matriz de 2x6 donde los valores de cada posición `(i, j)` correspondan a `y ^ 3 + 2j` para todo `i` par y `2i-j ^ 2` por todo `i` impar.

(0.5 puntos) NM

In [3]:

```
# Respuesta
```

Ejercicio 4

Considera la matriz [[1, 2, 3], [4, 5, 6]]. Transformala en la matriz 1D [1, 2, 3, 4, 5, 6].

Nota: Para resolver este ejercicio te pueden ser de utilidad las funciones ravel o flatten de Numpy que nos permite transformar un array 2D numpy a 1D.

(0.5 puntos) EG

In [4]:

```
# Respuesta
```

Ejercicio 5

Crea una matriz, Z, de tamaño 8x8 que vaya alternando los valores 0 y 1, siguiendo un patrón de tablero de damas.

Nota: Podéis comprobar el resultado con la función imshow de Matplotlib que nos permite visualizar una imagen 2D.

(1 punto) NM

In [5]:

```
# Respuesta
```

Matplotlib - Ejercicios

Ejercicio 1

Representa en un único gráfico las funciones f1 y f2 definidas más abajo, evaluadas en el intervalo [0, 2] y con intérvalo (resolución) 0.02. Haz que los rangos de valores mostrados sean `[0.25, 2.25]` per el eje X y `[-1, 1]` por el eje Y. Adicionalmente, muestra la gráfica correspondiente a `f1` con color verde i la gráfica correspondiente a `f2` con color azul.

(0.5 puntos) NM

In [6]:

```
import numpy as np
import matplotlib
```

```
import matplotlib
import matplotlib.pyplot as pl

def f1(x):
    return np.exp(-x) * np.cos(2 * np.pi * x)

def f2(x):
    return 1/(1 + np.exp(x))
```

In [7]:

```
# Respuesta
```

Ejercicio 2

Carga los datos del archivo `company_sales.csv`, que puedes encontrar en la carpeta `data`, en forma de *dataframe*. Este conjunto de datos recoge información sobre las ventas mensuales de una empresa de cosmética y productos de higiene.

Parte A **(0.5 puntos)** :

Muestra un gráfico de líneas de puntos de las ventas totales (*total_profit*) por cada mes (los meses están numerados 1 a 12).

Consideraciones:

- La línea debe ser con puntos (*dotted line*) de color rojo y de tamaño 3.
- Se debe mostrar la leyenda en la parte inferior derecha.

Parte B **(1 punto)** :

Muestra un gráfico circular (*pie chart*) con el número de unidades vendidas este año por producto en porcentaje.

Consideraciones:

- Debéis incluir el porcentaje (%) de cada producto dentro de su respectivo espacio del gráfico.
- Se debe mostrar la leyenda en la parte derecha.

Nota: En el Notebook de teoría hemos visto cómo mostrar gráficos lineares simples. Matplotlib nos ofrece muchos otros tipos de gráficos, siendo los gráficos circulares uno de ellos. Para resolver este ejercicio puedes consultar la documentación oficial de Matplotlib para ver cómo se trabaja con este tipo de gráficos.

(1.5 puntos) EG

In [8]:

```
# Respuesta
```

pandas - Ejercicios

Ejercicio 1

Carga los datos desde el archivo `netflixtitles.csv`, que puedes encontrar en la carpeta `data`, en un *dataframe*. Este conjunto de datos recoje información sobre películas y séries de TV de Netflix (https://www.kaggle.com/shivamb/netflix-shows) hasta el 2019.

Muestra el número de filas del *dataframe* y los nombres de las columnas.

Consideraciones:

- Muestra únicamente la información pedida en el enunciado.

(0.5 puntos) NM

In [9]:

```
# Respuesta
```

Ejercicio 2

Agrupa los datos cargados en el ejercicio 1 por **año** y, para cada año, muestra el nombre total de películas (*Movies*) y series de TV (*TV Show*) por separado.

Nota: Al Notebook de teoría hemos visto cómo calcular la media de una agrupación de datos. Para resolver este ejercicio necesitarás investigar cómo contar (*count*) todos los valores de una columna en una agrupación. Te puede ser de utilidad la función aggregate de pandas, la que nos permite aplicar diferentes funciones a una columna en un `groupby`.

(0.5 puntos) EG

In [10]:

```
# Respuesta
```

Ejercicio 3

Muestra las películas estrenadas el *2019* que tienen una duración superior a *100* minutos.

Nota: Al Notebook de teoría hemos visto operaciones básicas de filtrado. Para resolver este ejercicio necesitarás investigar cómo aplicar condiciones más complejas.

Nota 2: Los valores de la columna de duración (*duration*) son strings, ya que combinan números y letras. Por lo tanto, se tienen que transformar a int una vez se haya construido el subset de datos de películas. Te puede ser de utilidad la función astype de pandas.

(1 punto) EG

In [11]:

```
# Respuesta
```

Ejercicio 4

Queremos buscar aquellos países donde se estrenaron más de 10 películas al año. Debes mostrar por pantalla el **año**, el **nombre del país** y el **nombre total de películas** estrenadas en el año correspondiente.

Nota: Además del contenido de los notebooks de teoría, puedes utilizar lo que has aprendido haciendo el ejercicio 2.

(0.5 puntos) NM

In [12]:

```
# Respuesta
```

Ejercicio 5

Crea una copia del dataframe original añadiendo una nueva columna que nos indicará por cada fila la duración de la película. Definimos tres niveles:

- **short** : si la duración es inferior a 90 minutos.
- **medium** : si la duración está entre 90 y 150 minutos.
- **long** : si la duración es superior a 150 minutos.

Muestra por pantalla aquellas filas que correspondan a una duración corta (*short*).

Nota: pandas nos ofrece principalmente tres maneras de añadir nuevas columnas a un dataframe existente. Cualquiera de las tres formas son válidas para resolver este ejercicio. Podéis buscar en Internet cómo añadir columnas a un dataframe de pandas, ya que la documentación presenta los tres métodos de forma separada.

(1 punto) EG

In [13]:

```
# Respuesta
```

Ejercicio 6

En este ejercicio trabajaremos con el dataset de Netflix, pero analizando las series de TV. Muestra un gráfico de barras con el número de series de TV americanas agrupadas por categoría (*listed_in*).

Consideraciones:

- Debes mostrar los datos de manera horizontal (barras horizontales).
- El eje de las Y debe incluir las etiquetas de las categorías de las series de TV.
- Debe incluir un texto con el número de series por cada categoría del gráfico. El texto debería estar situado a la derecha de cada barra.

Nota: En el Notebook de teoría hemos visto cómo mostrar gráficos lineares simples con el matplotlib. Para este ejercicio explora las funcionalidades que presenta la librería pandas para generar gráficos a partir de dataframes. También pueden ser de utilidad las funciones text o annotate para mostrar el número de series al lado de cada barra.

(1.5 puntos) `EG`

In [14]:

```
# Respuesta
```

Ejercicio Opcional

Python es un lenguaje de programación muy utilizado en el procesamiento de imágenes. En este ejercicio veremos una aplicación de detección de elementos en una imagen de microscopía de escaneo (en inglés: Scanning Element Microscopy (SEM)).

En esta imagen se puede ver una muestra de vidrio (gris claro) con algunas burbujas (elementos negros) y granos de arena (gris oscuro). Queremos ser capaces de detectar los diferentes elementos de la muestra a partir de la imagen.

Una imagen se puede ver como una matriz bidimensional donde cada elemento de la matriz corresponde al valor de un pixel de la imagen. Por ejemplo, si miramos la matriz de la imagen que os proporcionamos, veremos valores que van de 0 a 255. Así pues, para distinguir los diferentes elementos de la muestra de vidrio, nos tendremos que fijar en el nivel de gris de los píxeles de la imagen a través del histograma. `EI`

Parte A:

- Para poder identificar mejor los elementos de la muestra de vidrio, primero hemos aplicado un filtro de mediana para reducir el ruido. Grafica ahora la distribución de los píxeles de la imagen inicial y la imagen filtrada. ¿Qué diferencia puedes apreciar entre ambas distribuciones?

Nota: Utiliza la función distplot de la librería Seaborn para visualizar la distribución de los valores píxeles de la imagen.

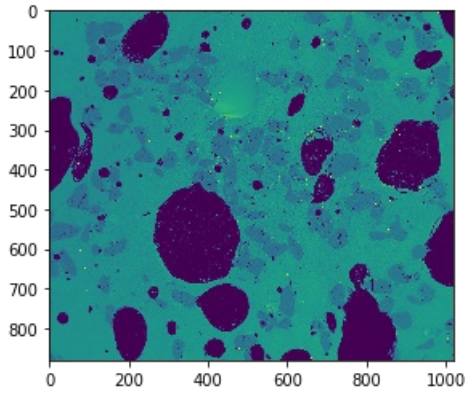
Parte B:

- Utilizando los valores del distribution plot de la imagen filtrada, determina unos umbrales que permitan diferenciar los píxeles de arena, los píxeles de vidrio y los píxeles de burbuja.
- Una vez que hayas definido los umbrales, muestra por pantalla una imagen donde cada uno de los elementos esté en un color diferente.

In [15]:

```
# Cargamos las librerías necesarias

import numpy as np
import matplotlib.pyplot as plt

# Librería de procesamiento de imágenes
from scipy import ndimage

# Librería de herramientas de visualización, similar al matplotlib
import seaborn as sns

# Leemos la imagen
data = plt.imread('data/SEM_image.jpg')

# Mostramos la imagen por pantalla
```

```
plt.imshow(data)

# Aplicamos un filtro de mediana para reducir el ruido de la imagen
filtdata = ndimage.median_filter(data, size=(7,7))
```



In [16]:

```
# Respuesta
```