**Indice**

1. **[Introduccion](#_Introduccion)**
2. **[Funciones, Librerias y Pandas basico](#_Funciones, Librerias y Pandas basico)**
3. **[Pandas](#_Pandas)**
4. **[Data Visualization](#_Data Visualization)**
5. **[Machine Learning](#_Machine Learning)**
6. **[Machine Learning: Optimizacion de modelos](#_Machine Learning: Optimizacion de modelos)**
7. **[Oracle ADS](#_Oracle ADS)**
8. **[Machine Learning con Oracle ADS](#_Machine Leaning con Oracle ADS)**

# Introduccion



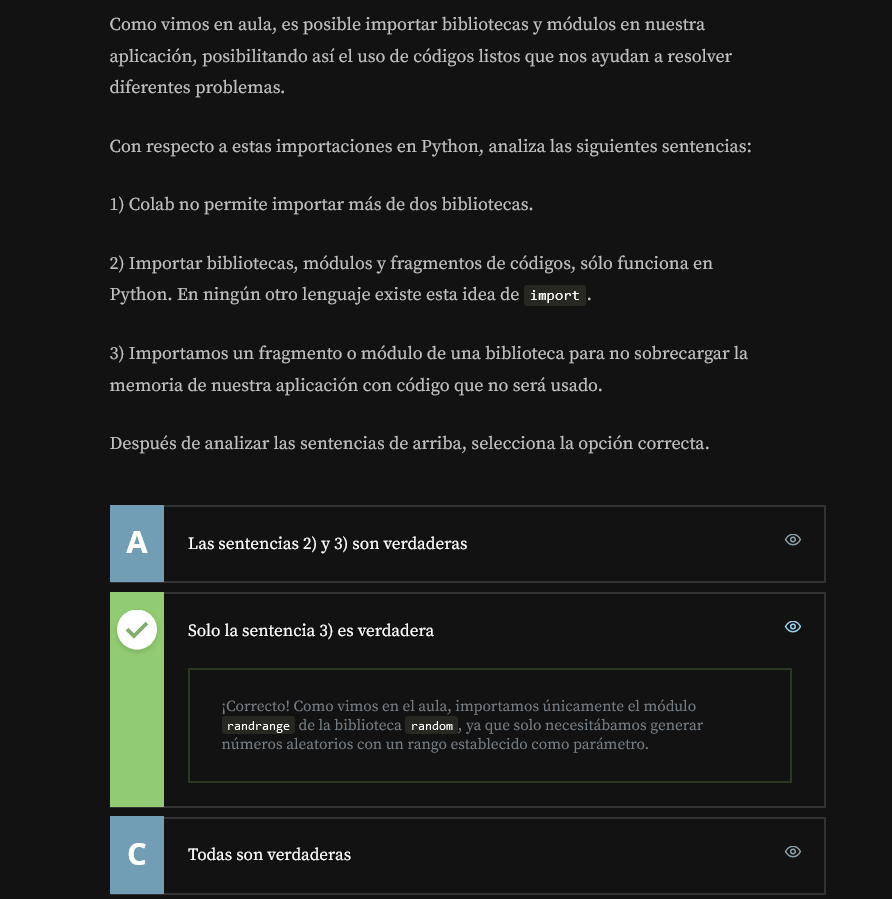


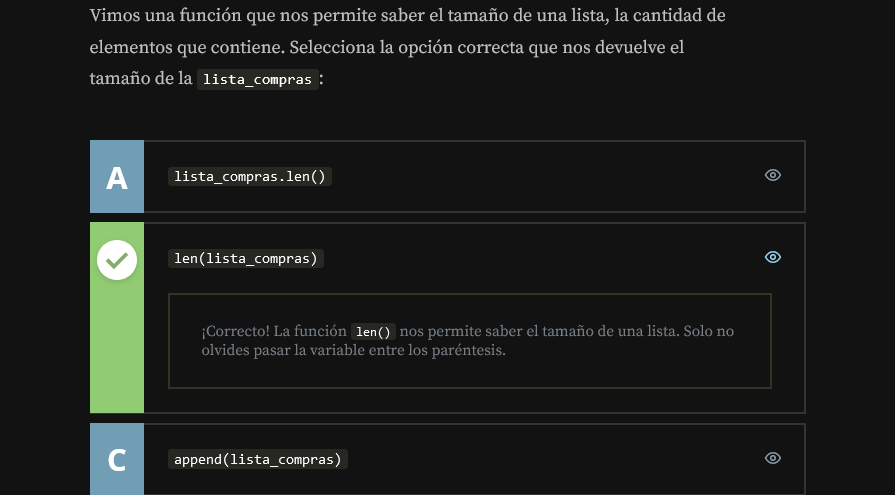


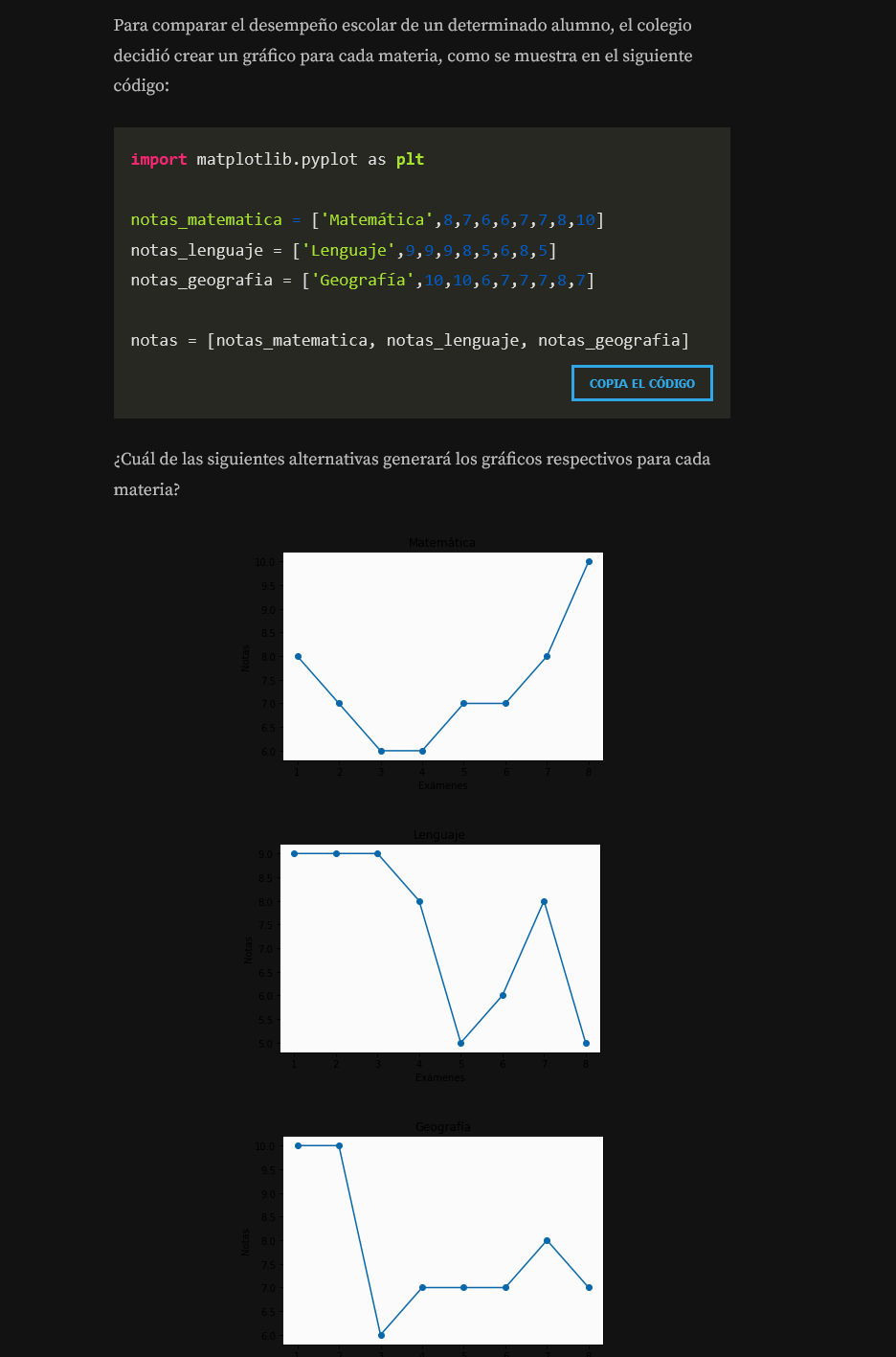


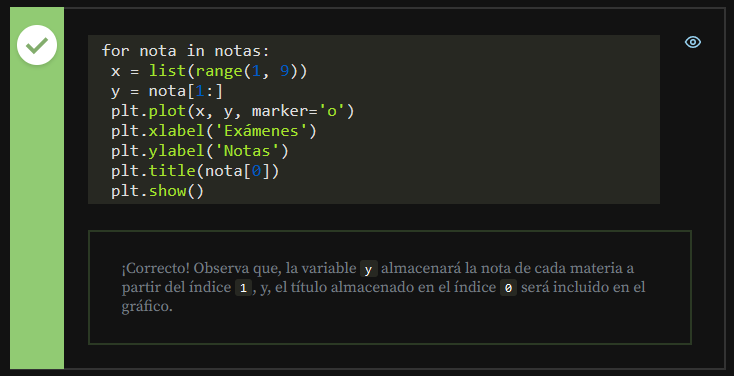






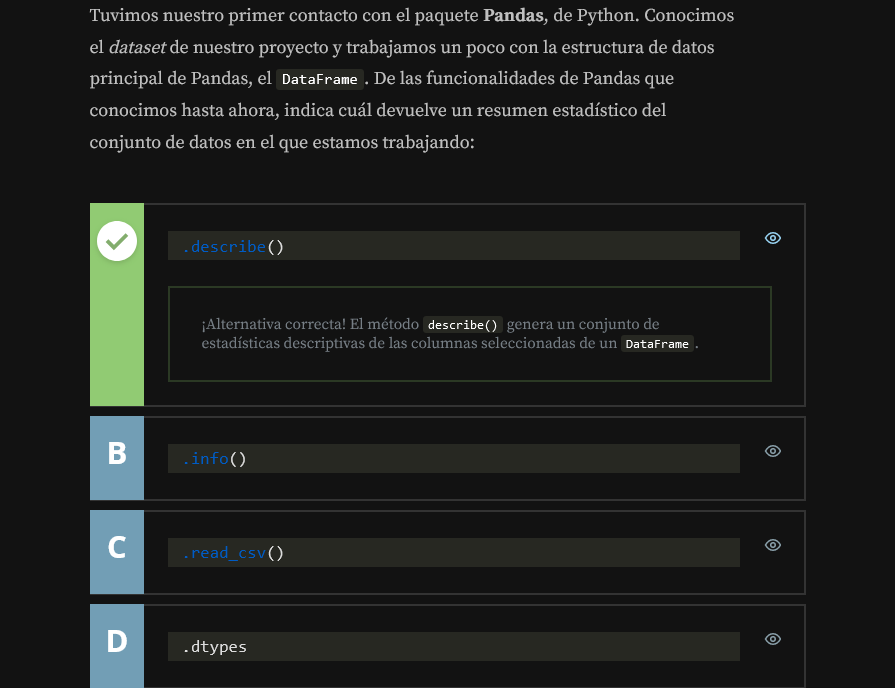




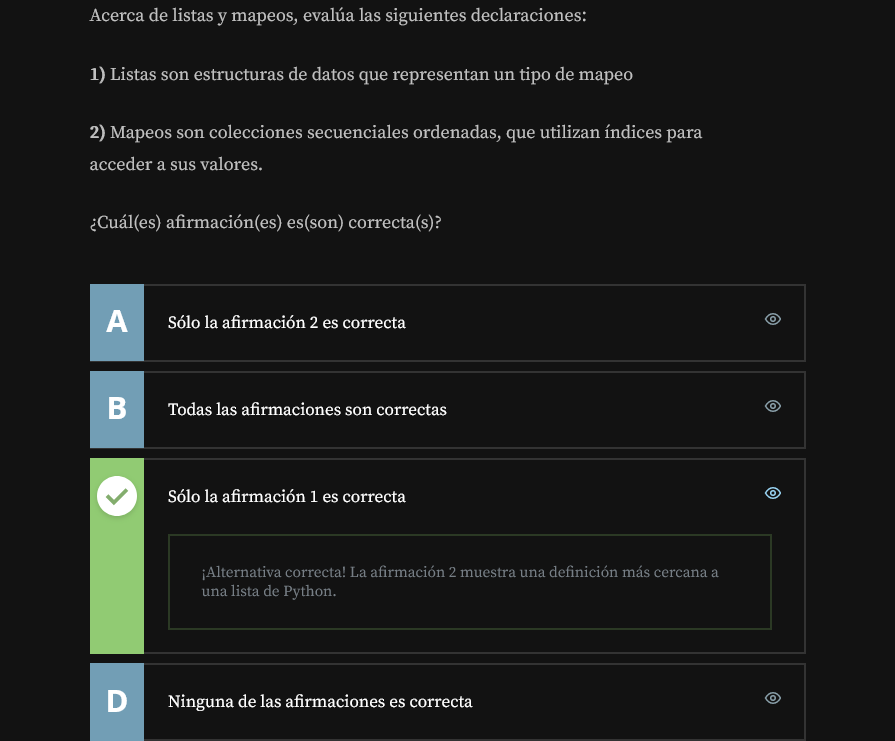


# Funciones, Librerias y Pandas basico



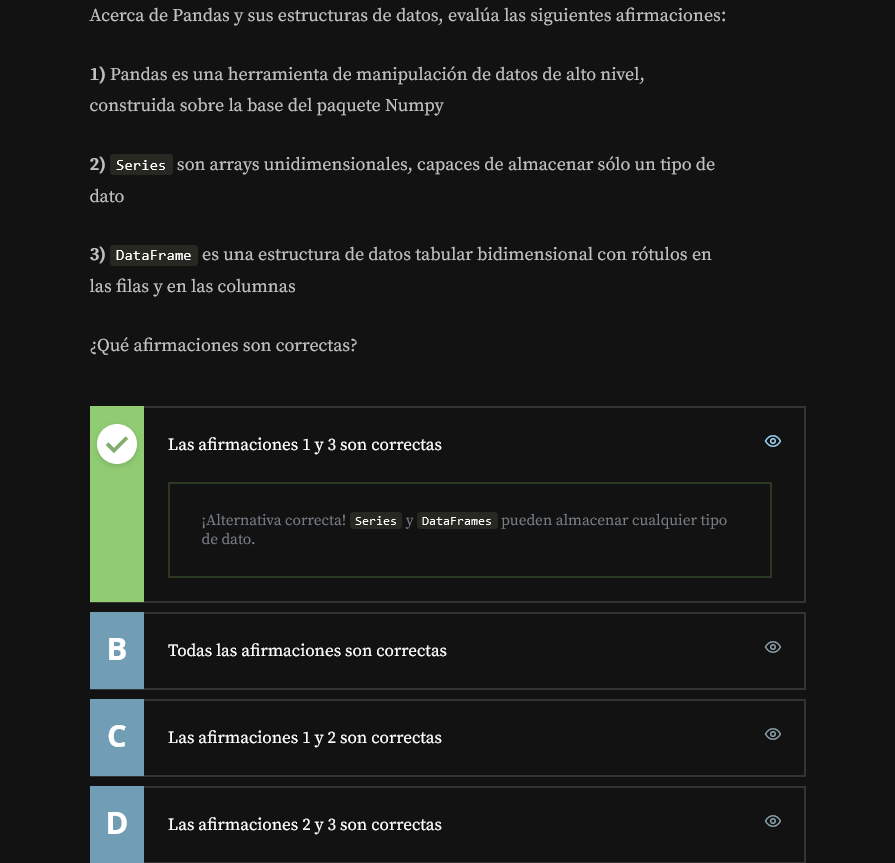


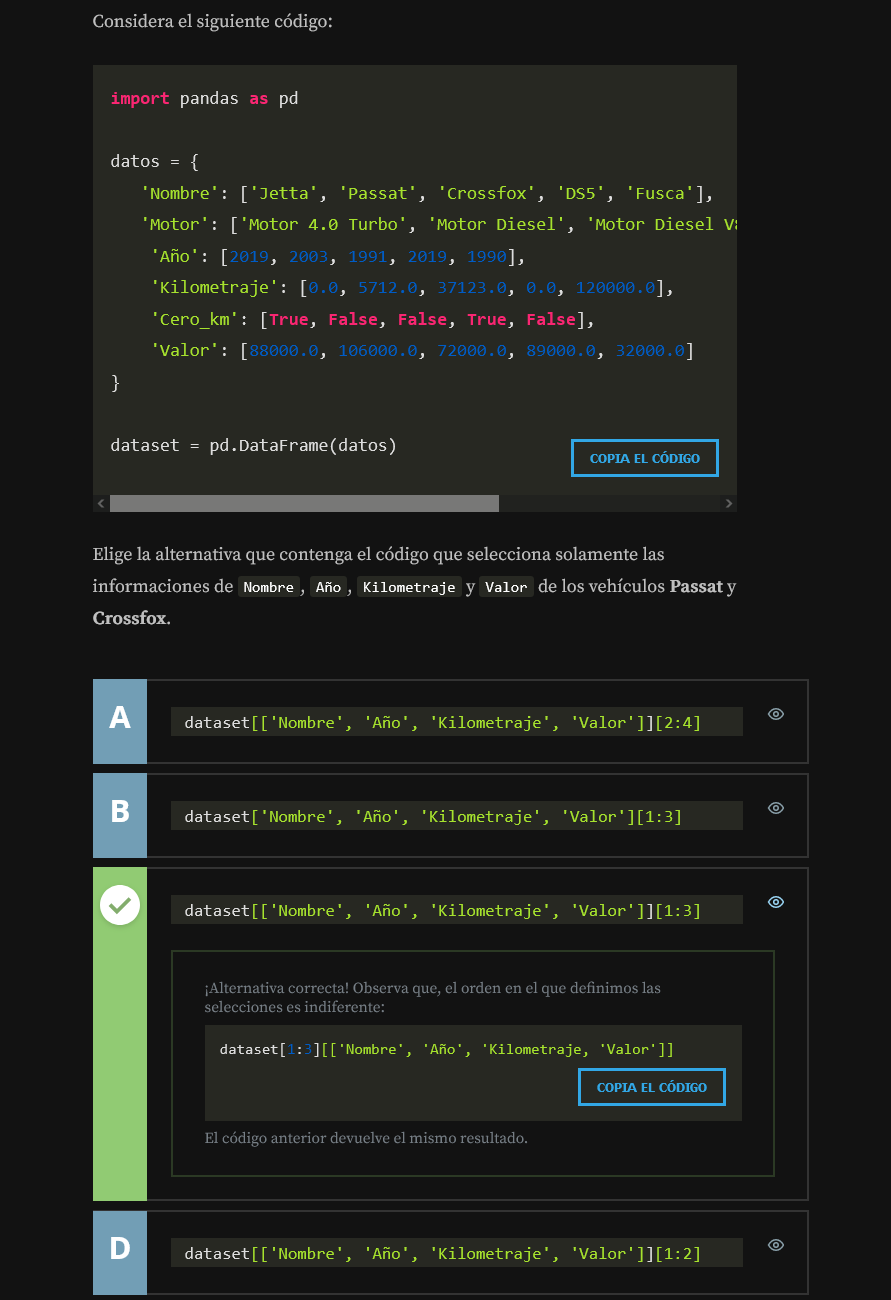


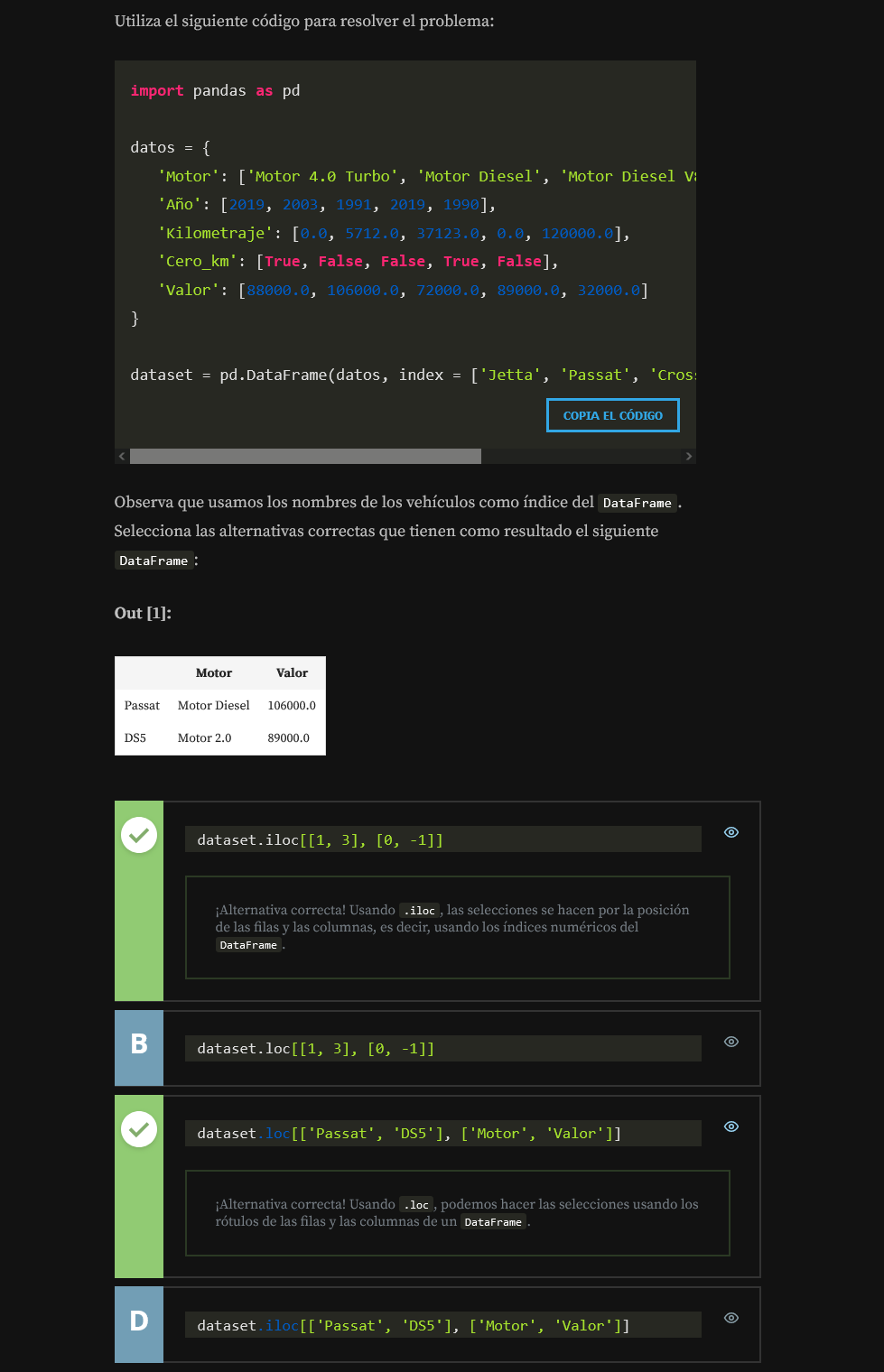


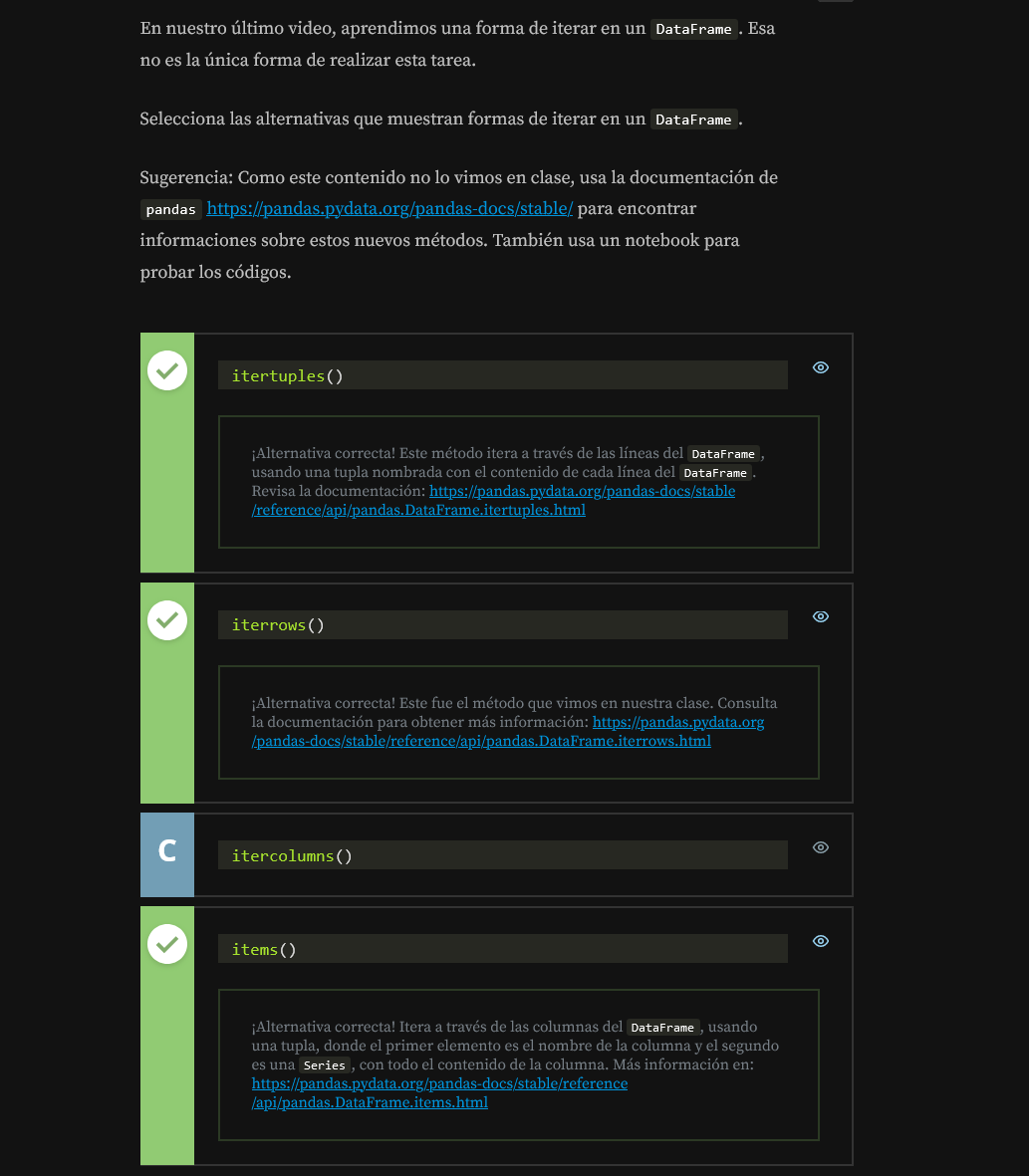










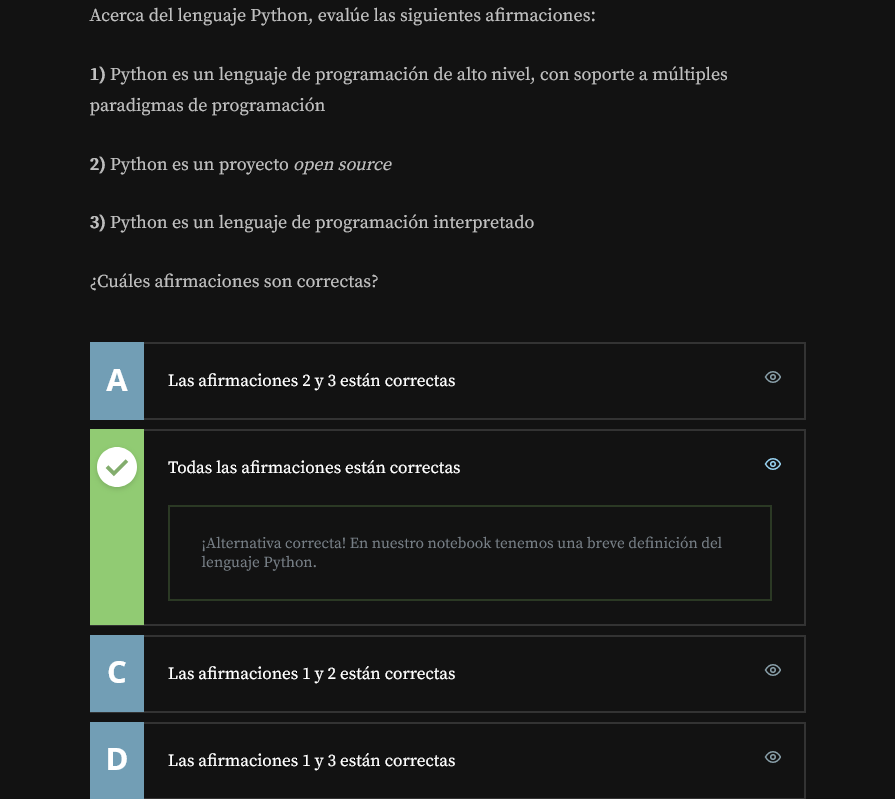


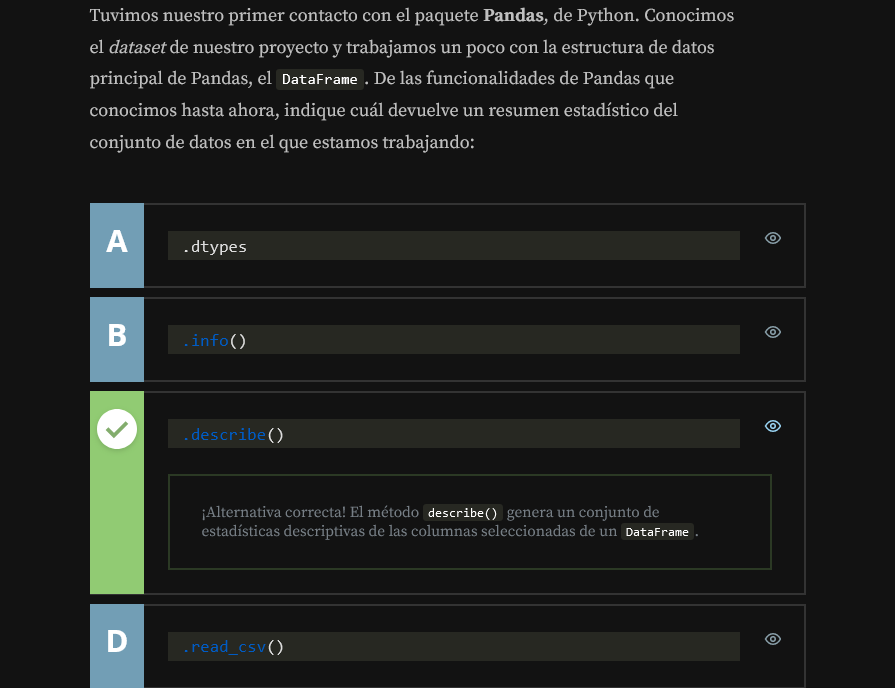
<https://www.aluracursos.com/blog/clasificando-texto-con-python>

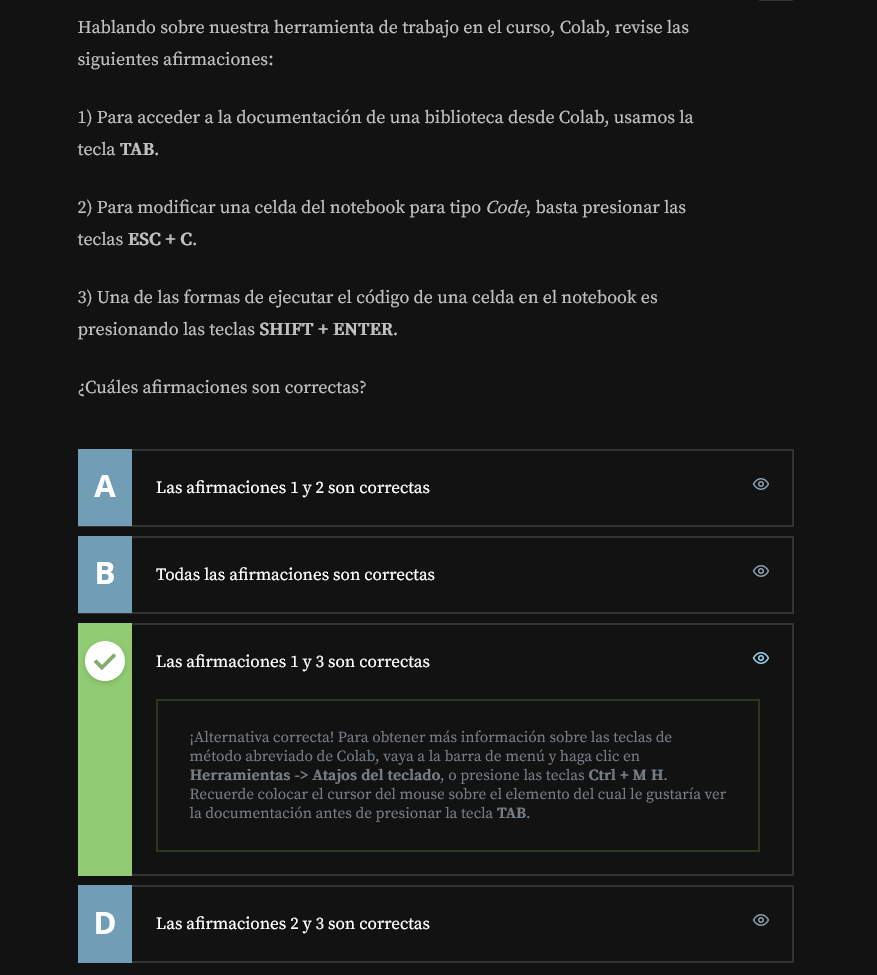
<https://www.aluracursos.com/blog/como-comparar-objetos-en-python>

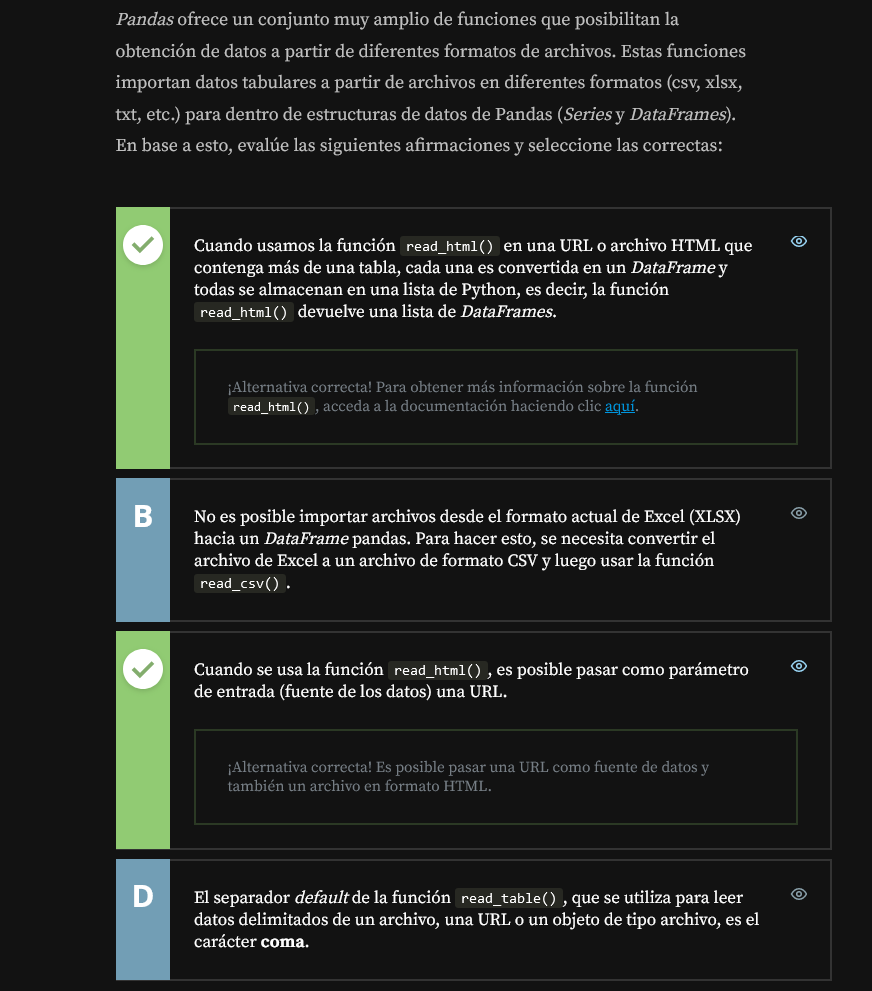
<https://www.aluracursos.com/blog/buscando-tweets-con-python>

# Pandas



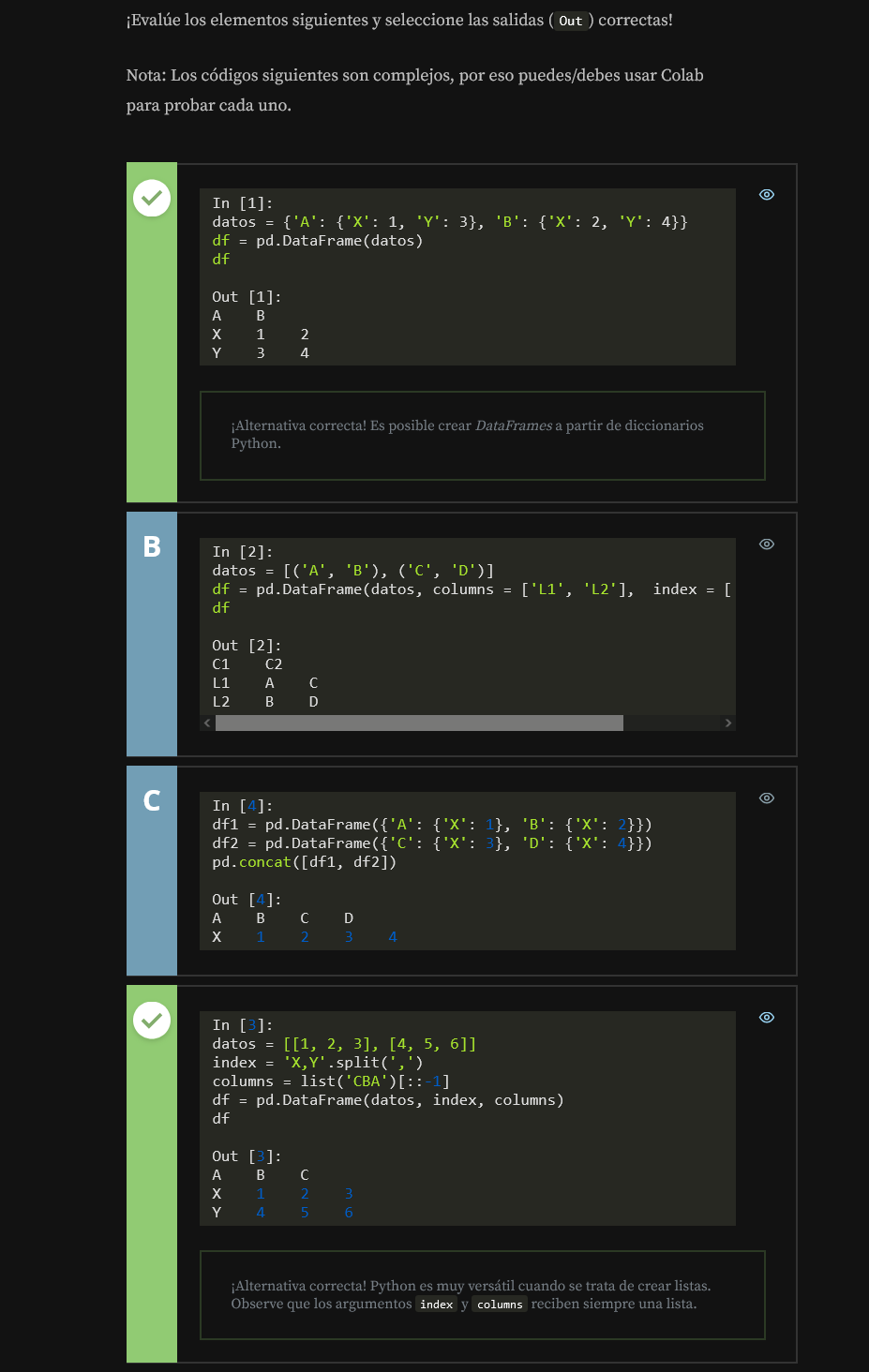


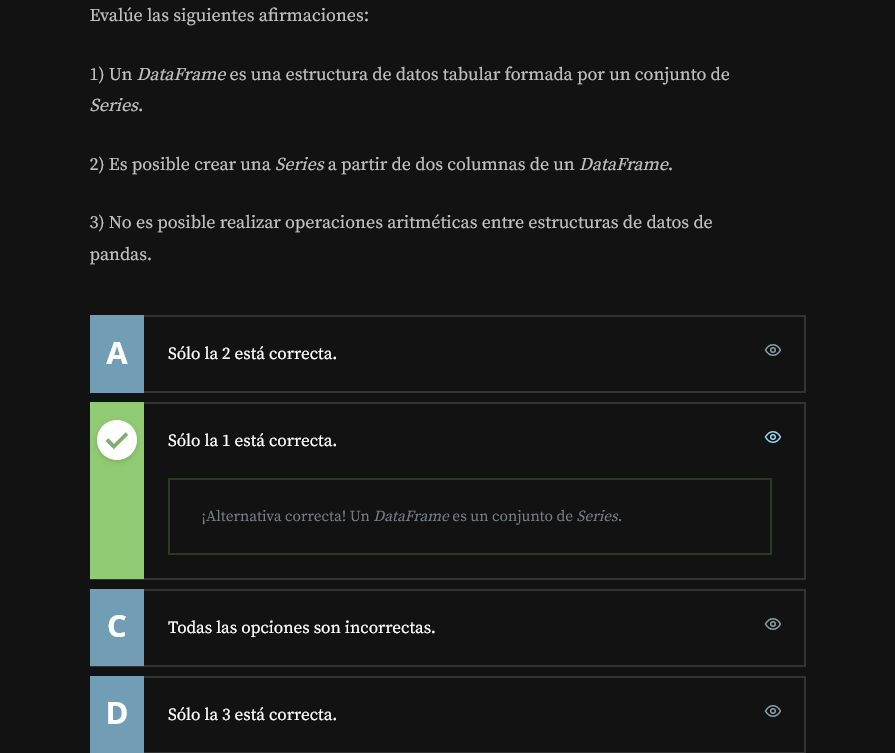


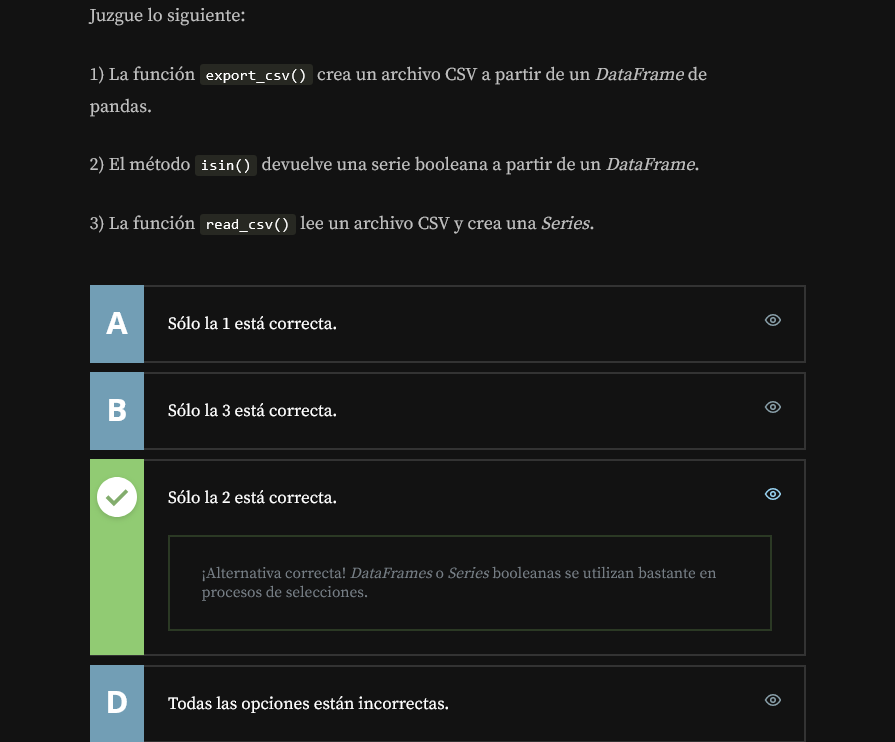


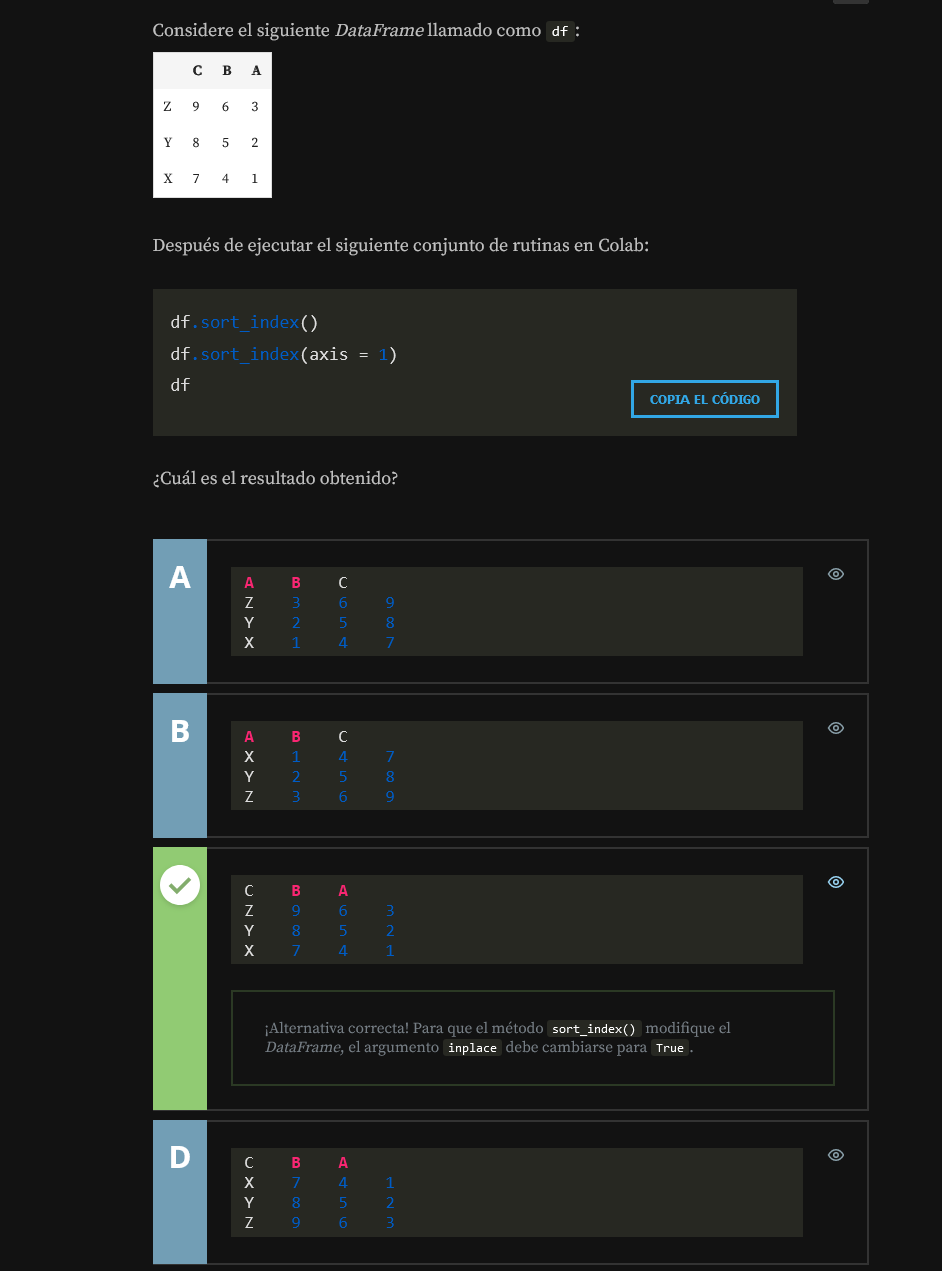


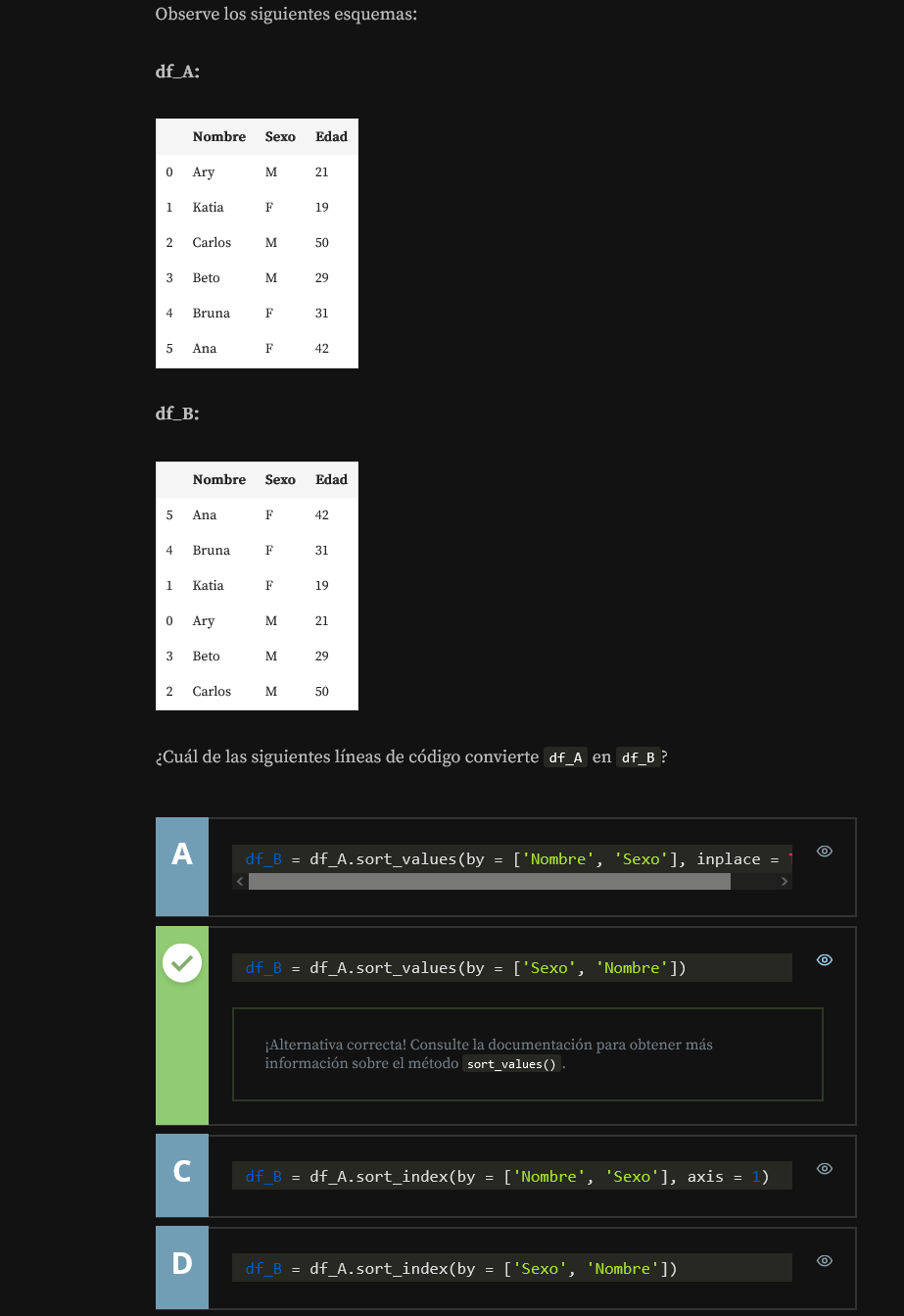


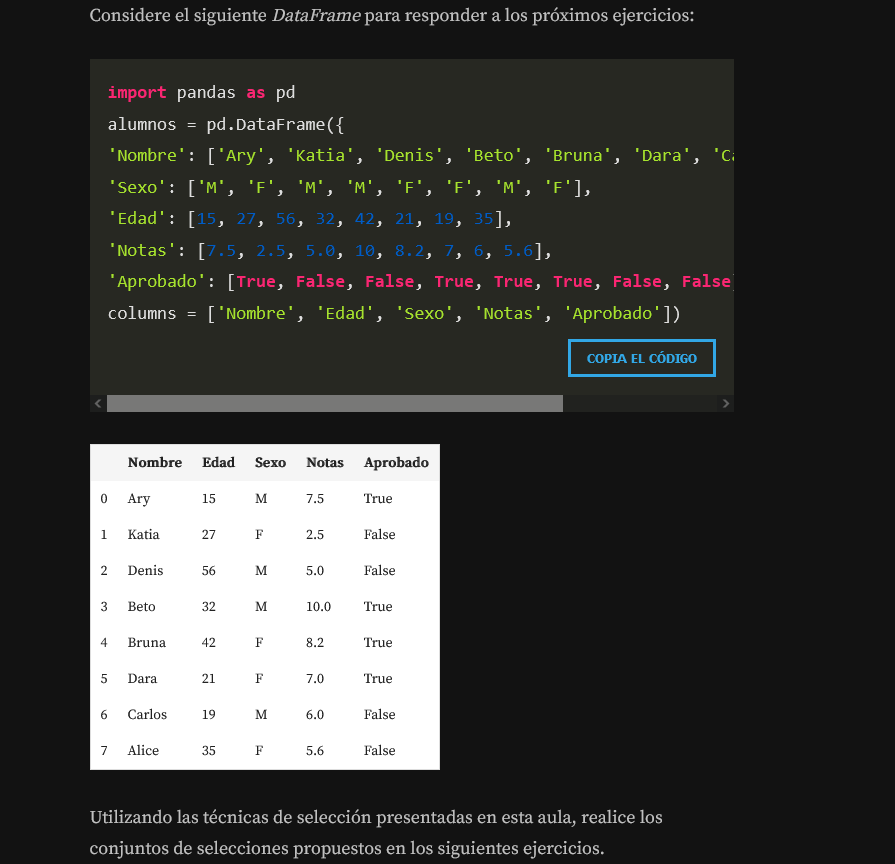


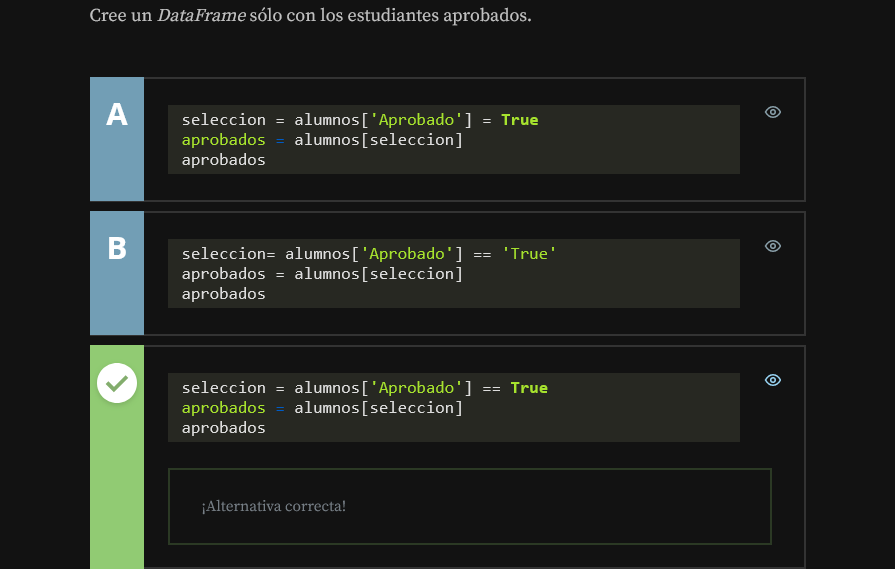


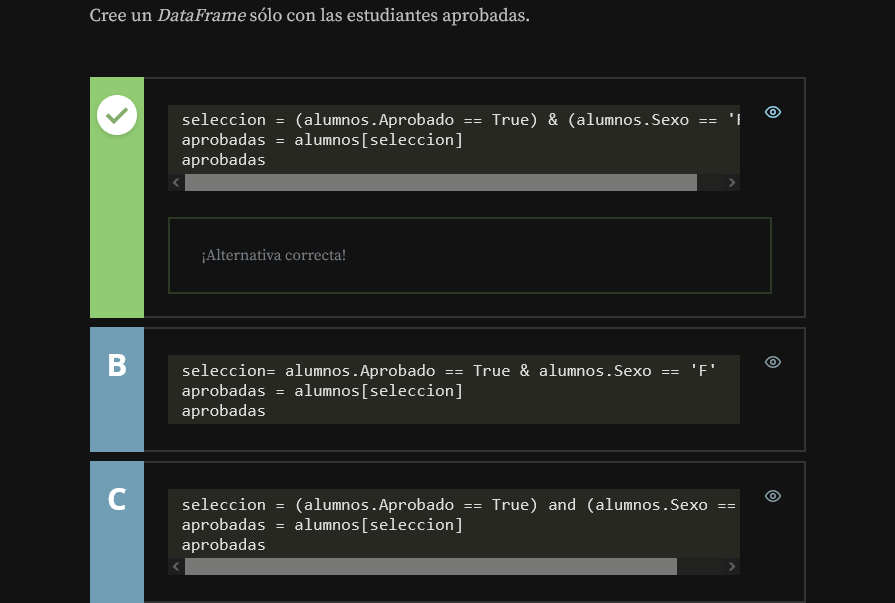




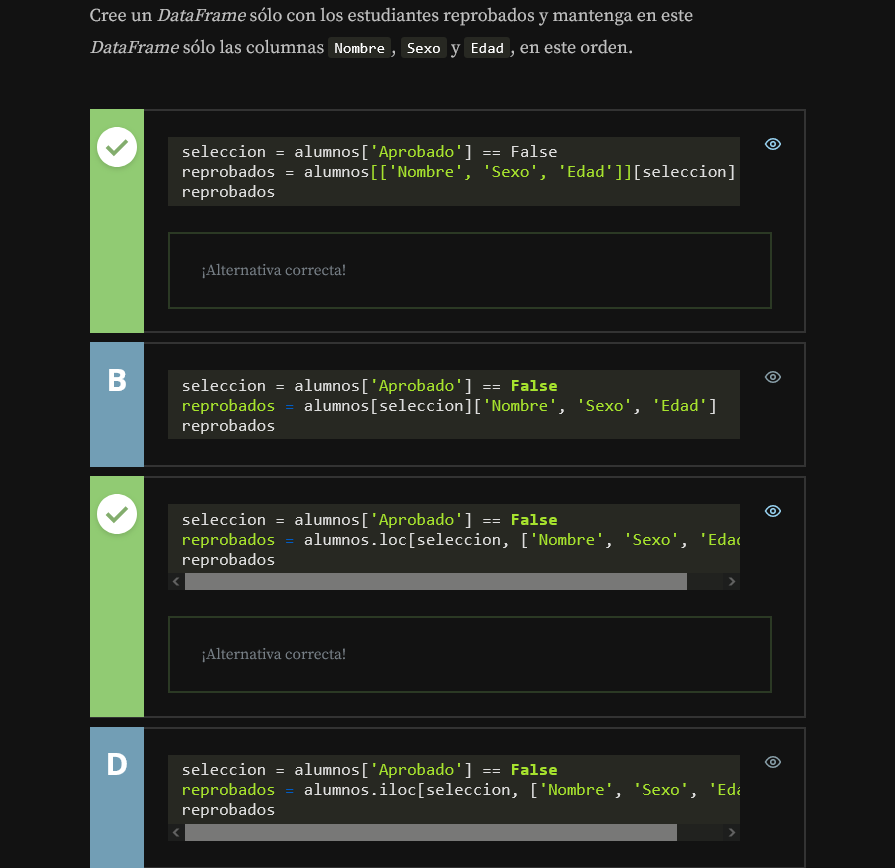


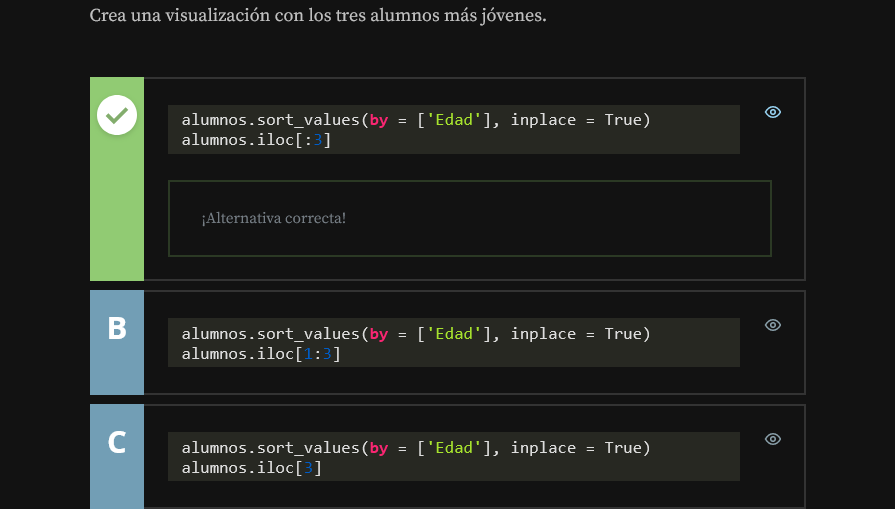




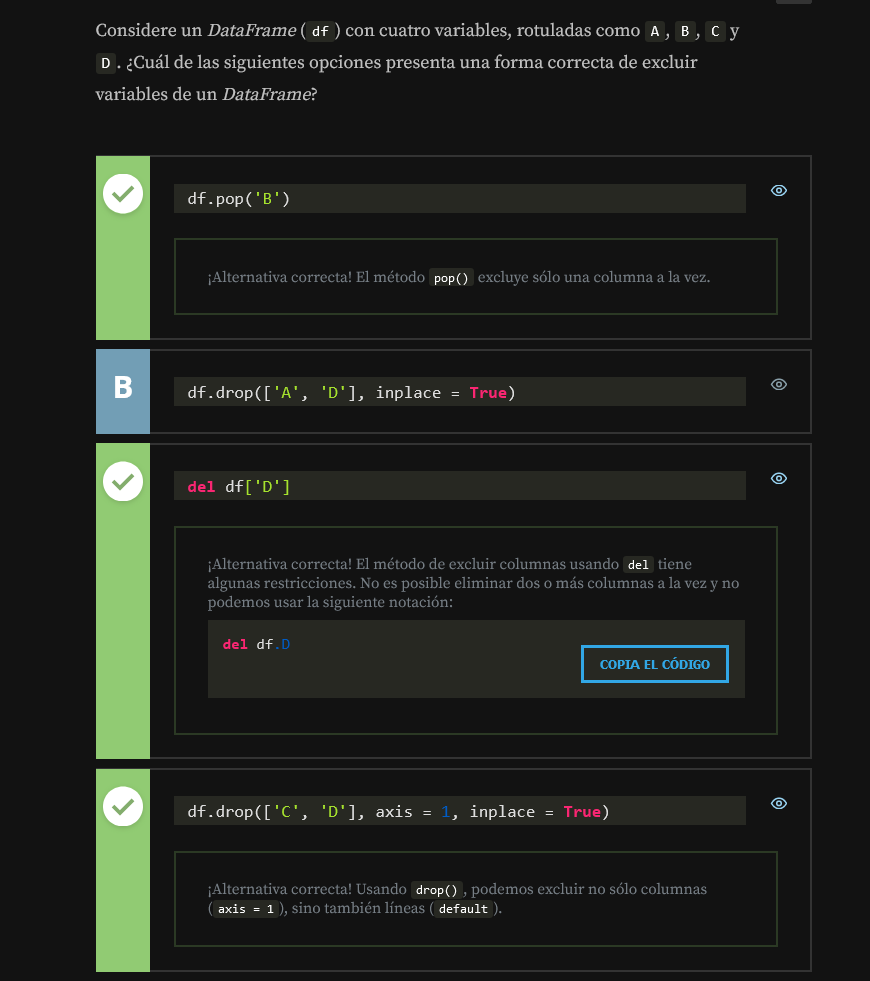




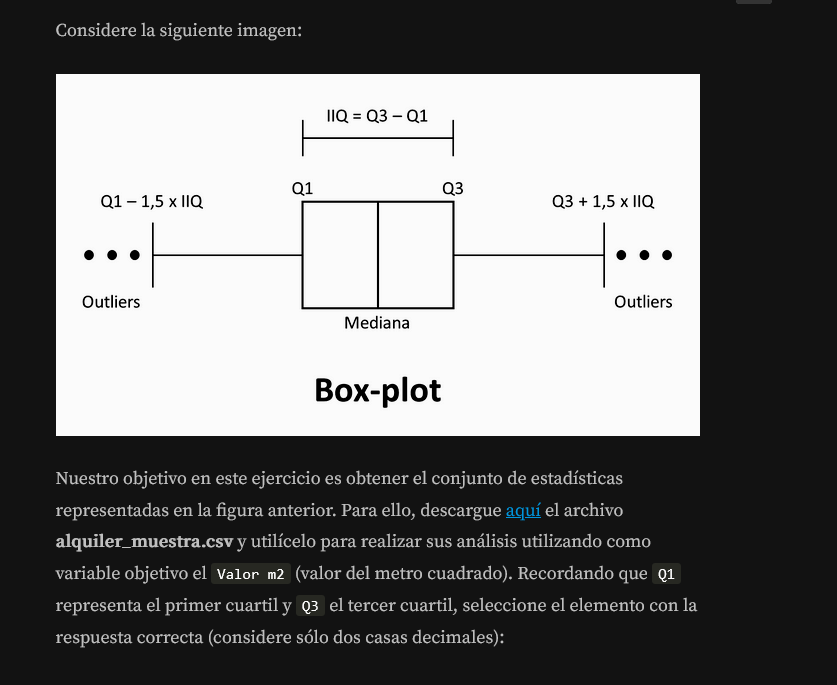


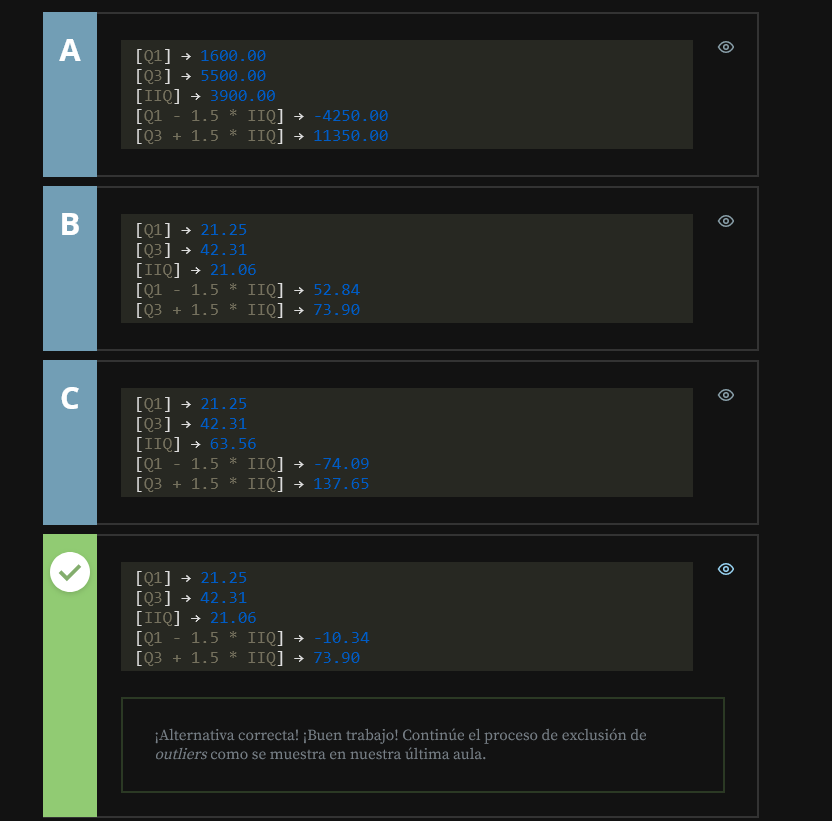






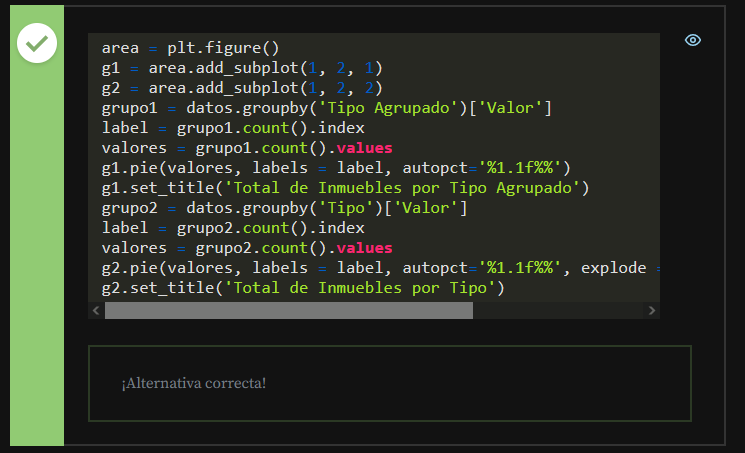








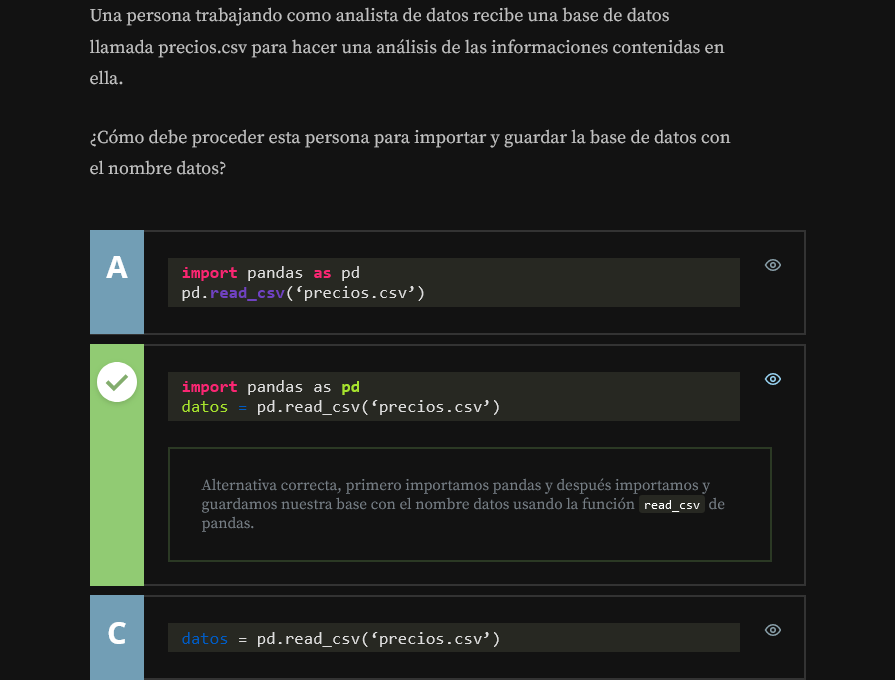


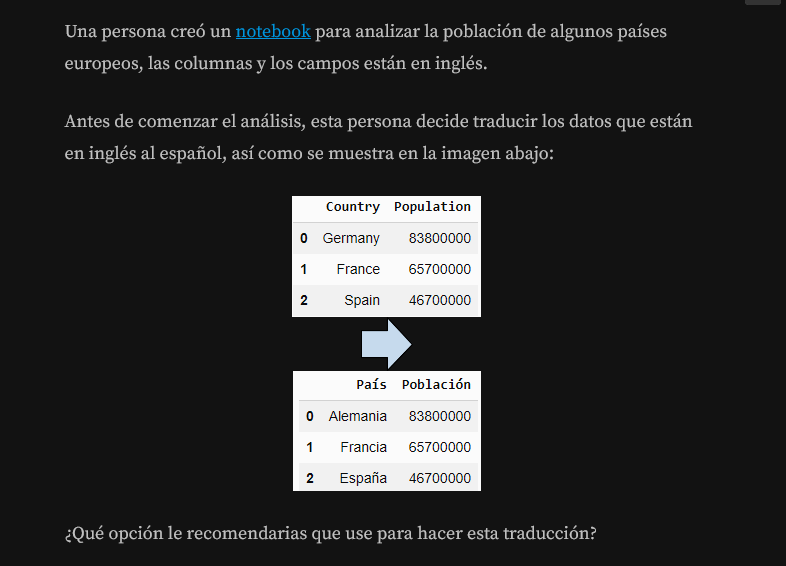


<https://www.aluracursos.com/blog/manipulando-datos-gigantes-con-pandas>

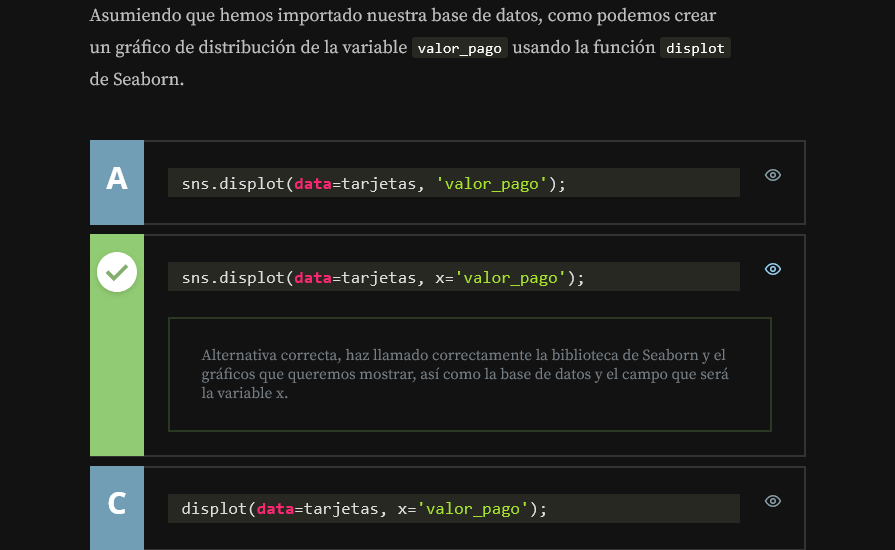
<https://www.aluracursos.com/blog/como-eliminar-filas-y-columnas-en-pandas>

# Data Visualization

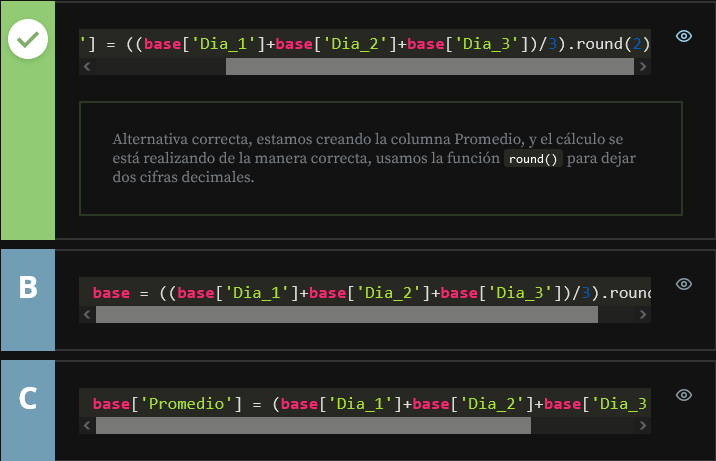


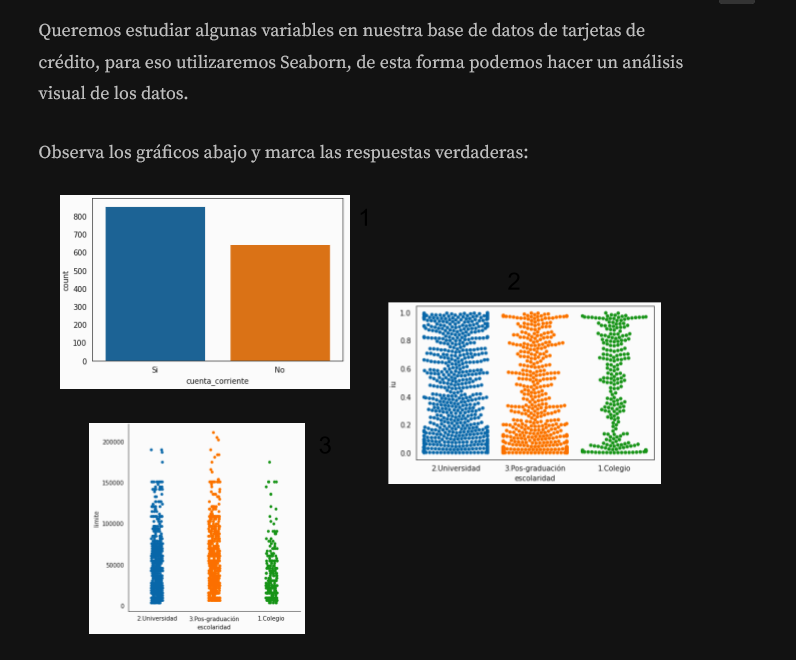


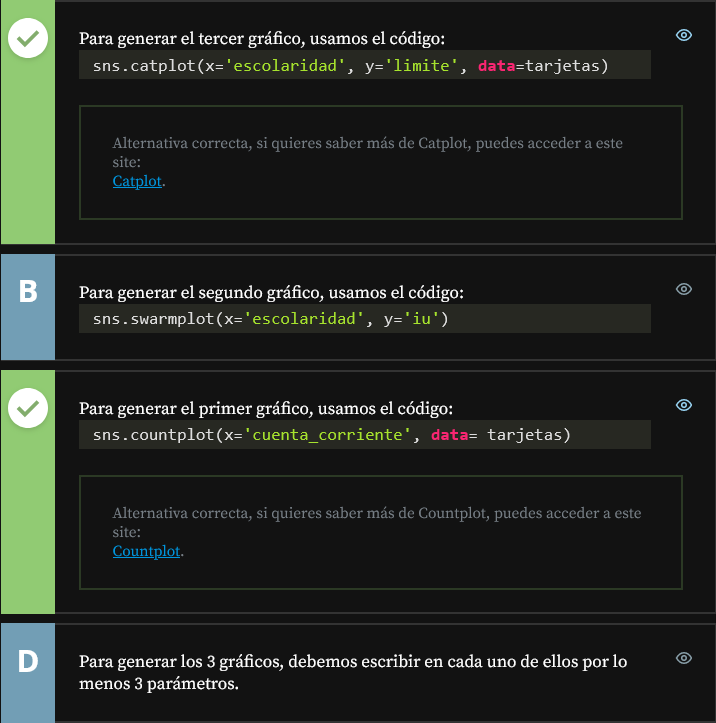




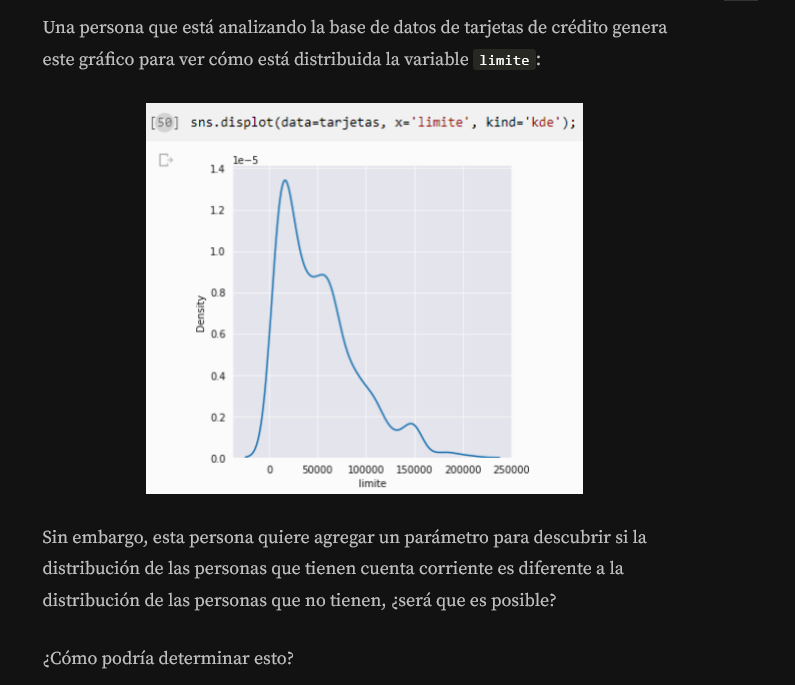


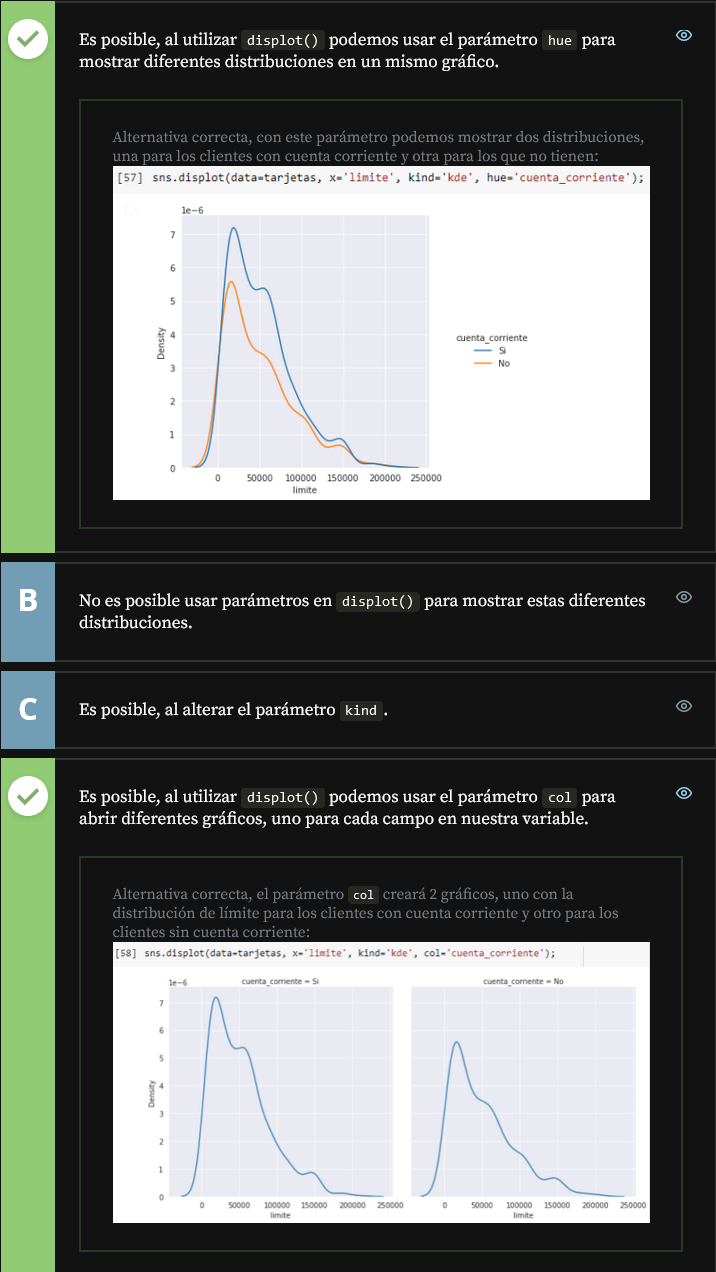




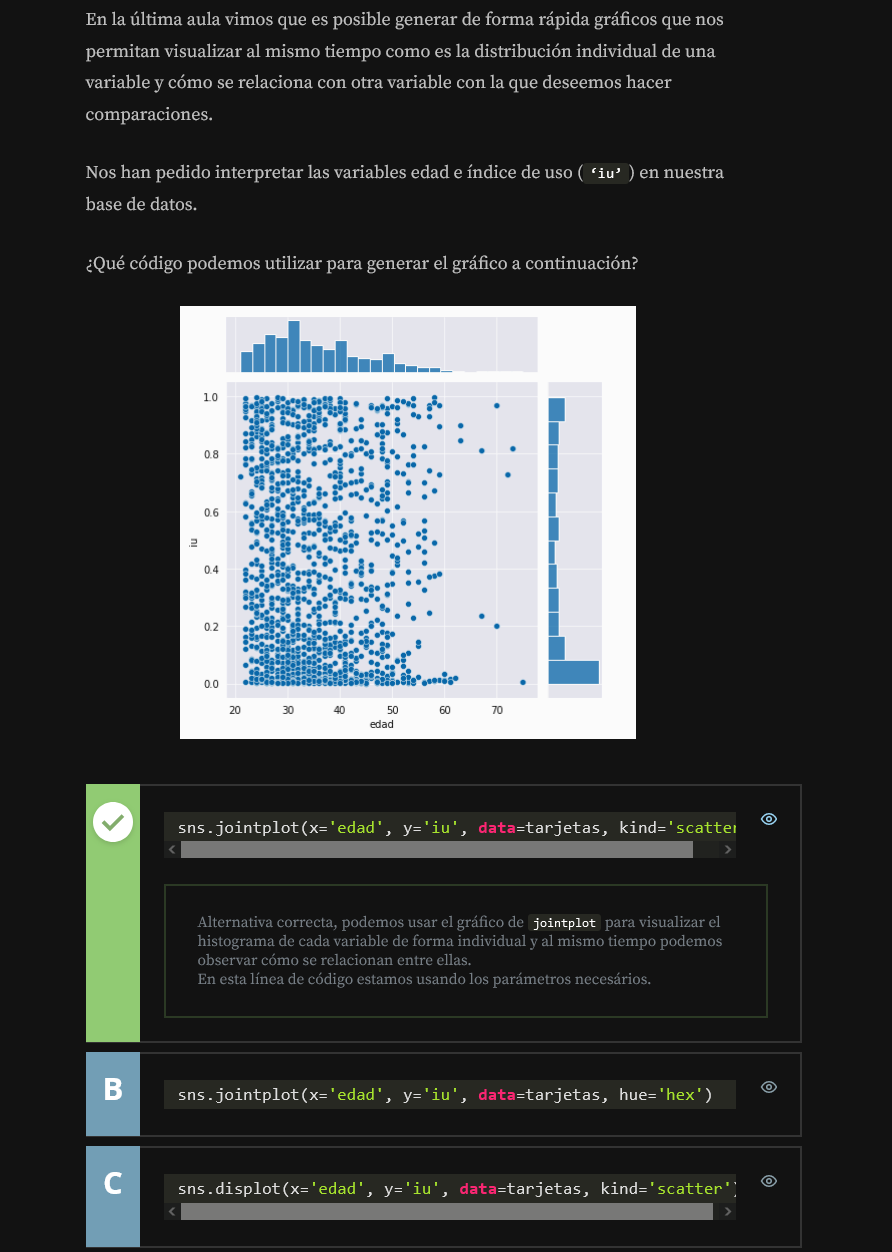


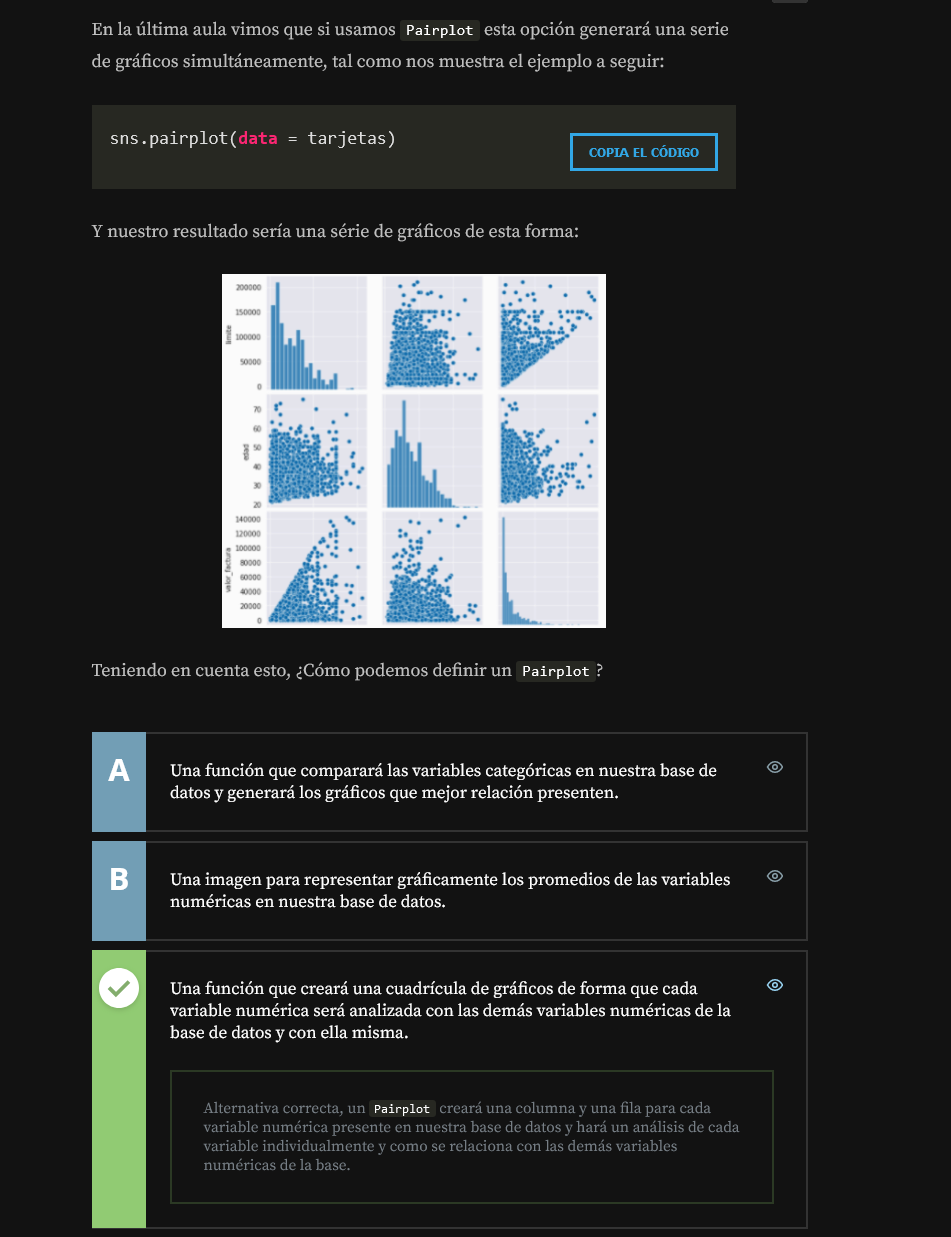








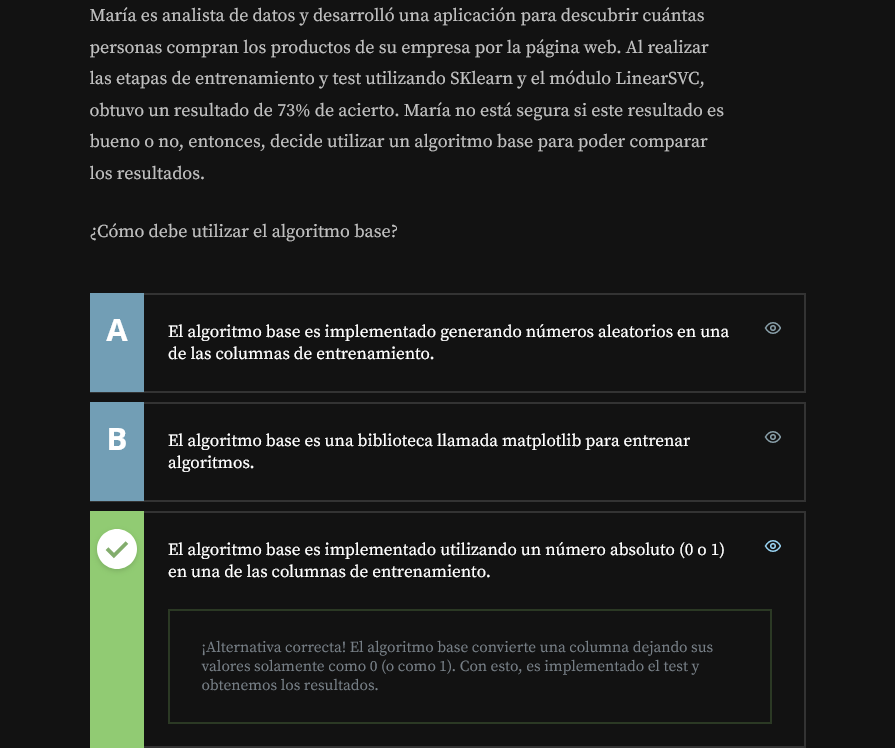




# Machine Learning







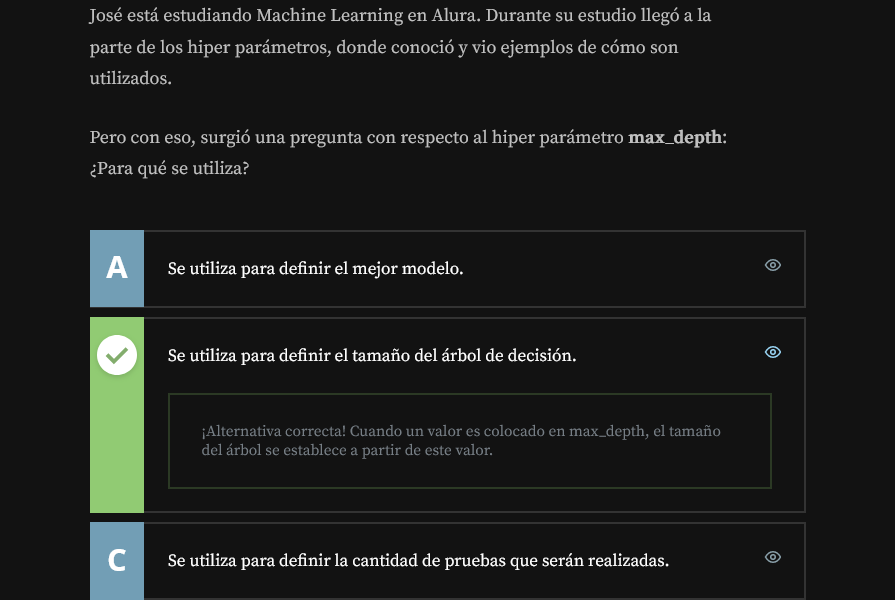




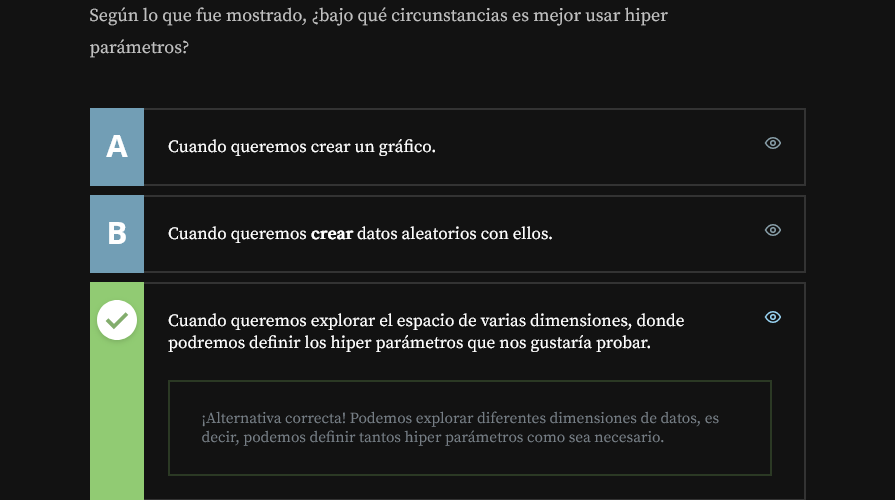
<https://github.com/alura-es-cursos/1918-machine-learning-clasificacion-con-sklearn/blob/proyecto-final/ML_clasificacion_con_SKLearn.ipynb>

<https://www.aluracursos.com/blog/primeros-pasos-en-inteligencia-artificial-ia>

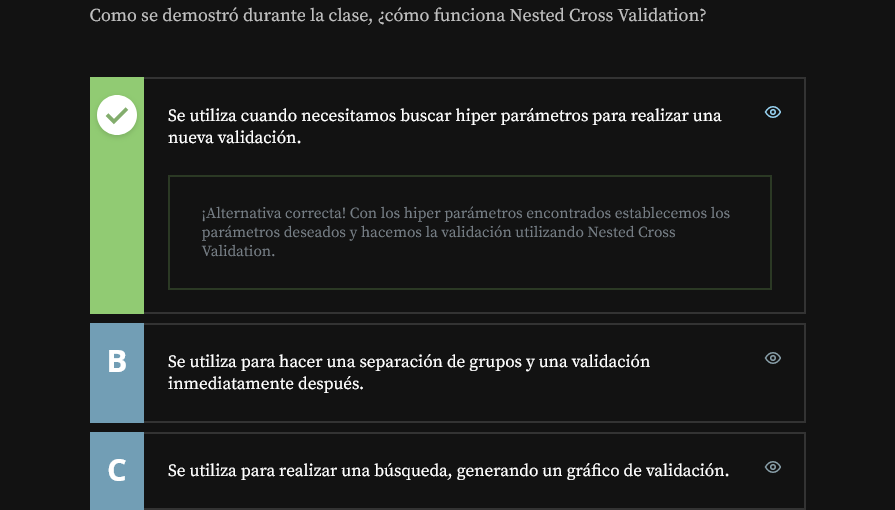
# Machine Learning: Optimizacion de modelos



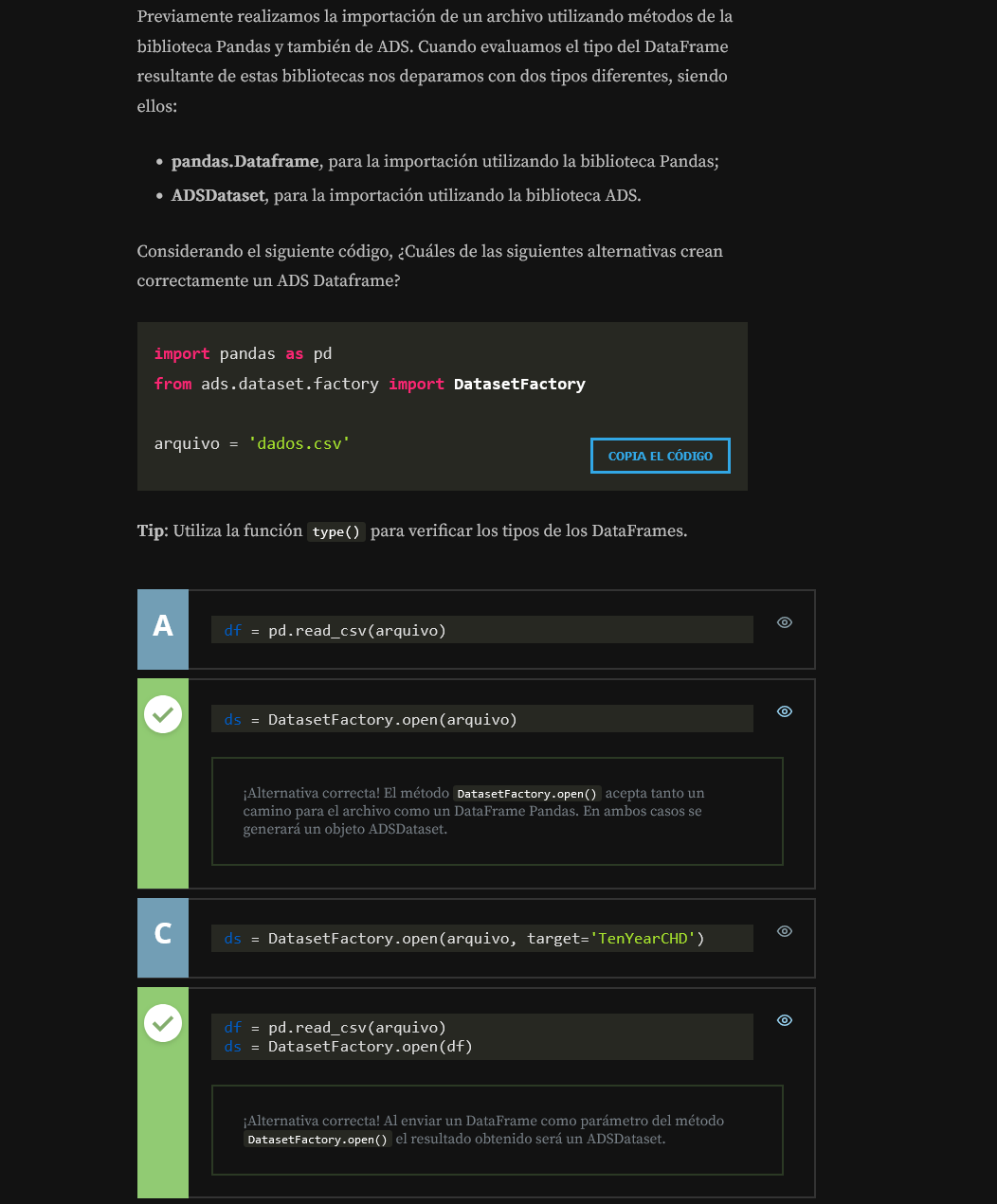








# Oracle ADS

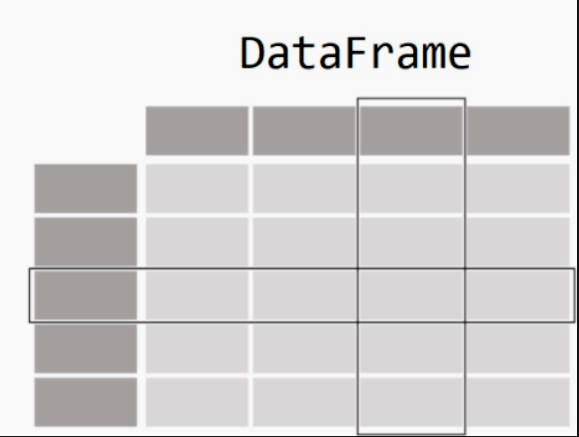


Existen diferentes tipos de DataFrames; el más conocido de ellos es el de la biblioteca **Pandas**. Pero, además de este, tenemos otros tipos pertenecientes a otras bibliotecas, como es el caso del **ADS DataFrame** que aprendimos en los últimos videos. Cada uno de estos tipos poseen sus características específicas y pueden ser utilizados en diferentes situaciones.

Vamos a aprender algunas características adicionales de estos DataFrames y también a conocer un poco sobre la biblioteca **Dask**.

* **Pandas**

Pandas es una biblioteca escrita en Python, utilizada para el análisis y la manipulación de datos. El **Pandas DataFrame** es una estructura de datos bidimensional que puede almacenar datos de diferentes tipos (enteros, floats, datos categóricos, strings, entre otros) en columnas. Su formato es similar al de una planilla de Excel, siendo compuesto por filas y columnas.



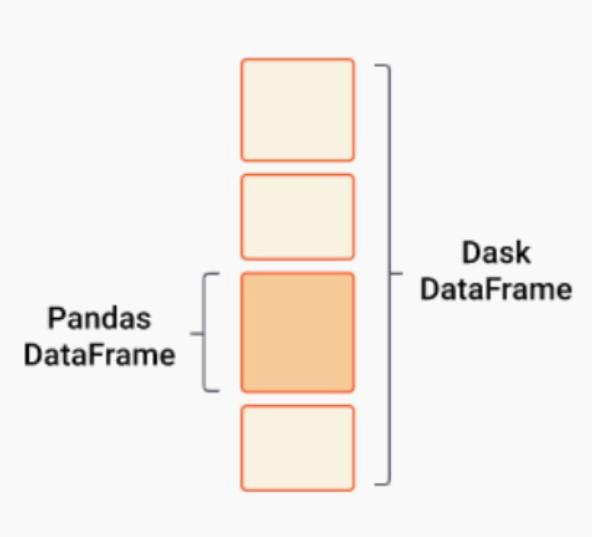
Las principales características de este DataFrame son:

* Fácil aprendizaje;
* Almacena diferentes tipos de datos;
* Tratamiento flexible y simple de datos faltantes;
* Acepta algunas operaciones aritméticas;
* Fácil manipulación.

Sin embargo, el uso de esta biblioteca no es muy indicado para trabajar con un gran volumen de datos. Esto se debe a que Pandas necesita que todos los datos estén prontamente disponibles en la memoria de tu computador y ello genera una limitación de la cantidad de datos con los cuales podemos trabajar y exigiendo mucho de nuestra máquina.

* **Dask**

Dask es una biblioteca creada para la computación paralela en Python, ella funciona bien con otras bibliotecas como **Pandas**, **Numpy**, **Scikit-learn**, entre otras. El **Dask DataFrame** se compone de muchos **Pandas DataFrames**, lo que hace que la manera como el computador trabaja con los comandos en este tipo de DataFrame sea diferente.



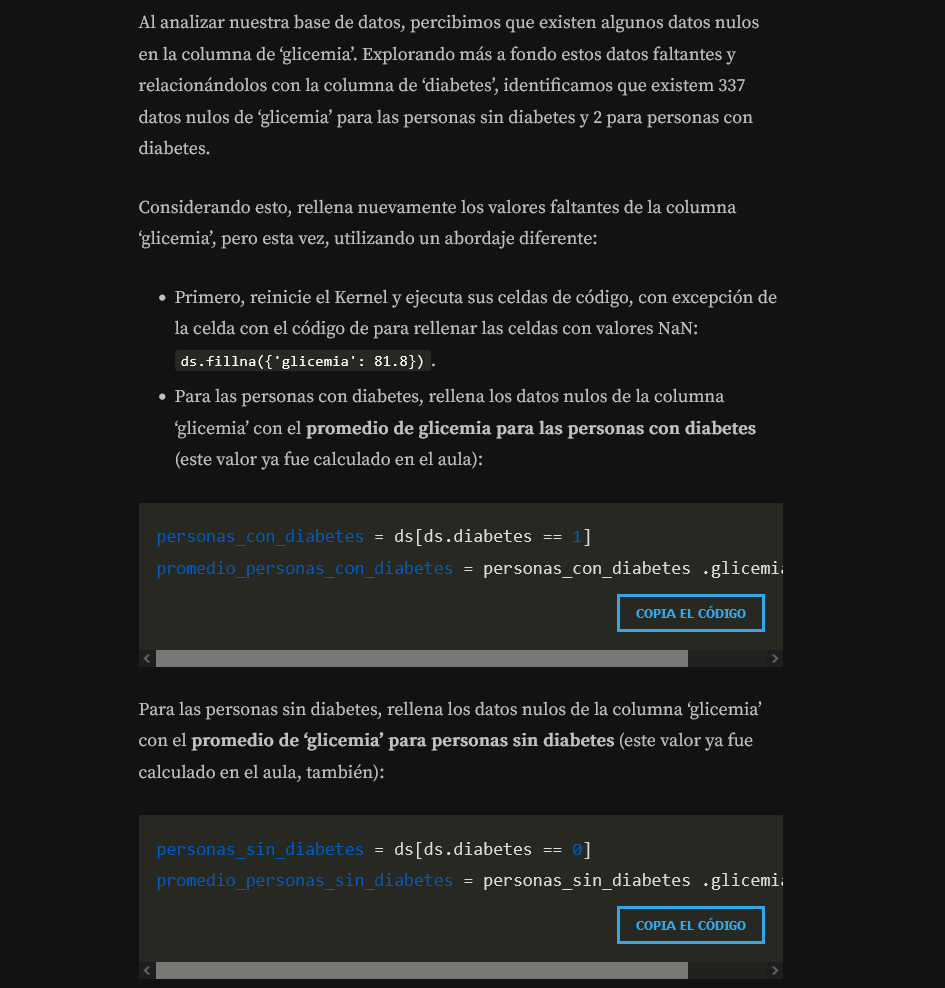
La forma diferente de trabajo de Dask se da por una característica de este en atrasar tareas, en la cual, él prepara un conjunto de transformaciones o cálculos para una ejecución posterior en paralelo. Esta estructura es diferente de otras funciones en Python que son computadas instantáneamente, al momento en que son llamadas.

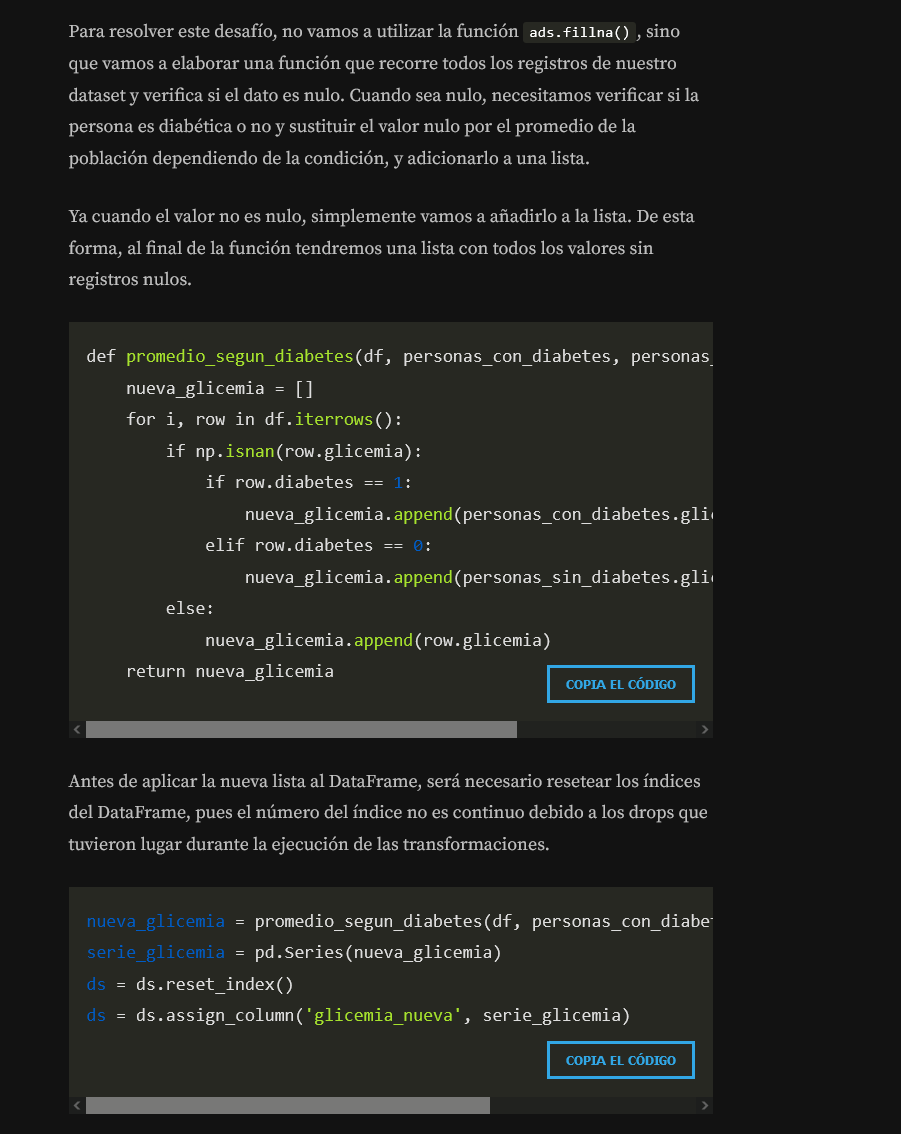
De esta forma, el atraso de tareas de Dask permite que esta biblioteca lidie muy bien con un conjunto de datos muy grande o con muchos grupos en sus datos, lo cual es una ventaja sobre Pandas, que necesitaría de un poder de procesamiento muy alto para lidiar con datos como estos.

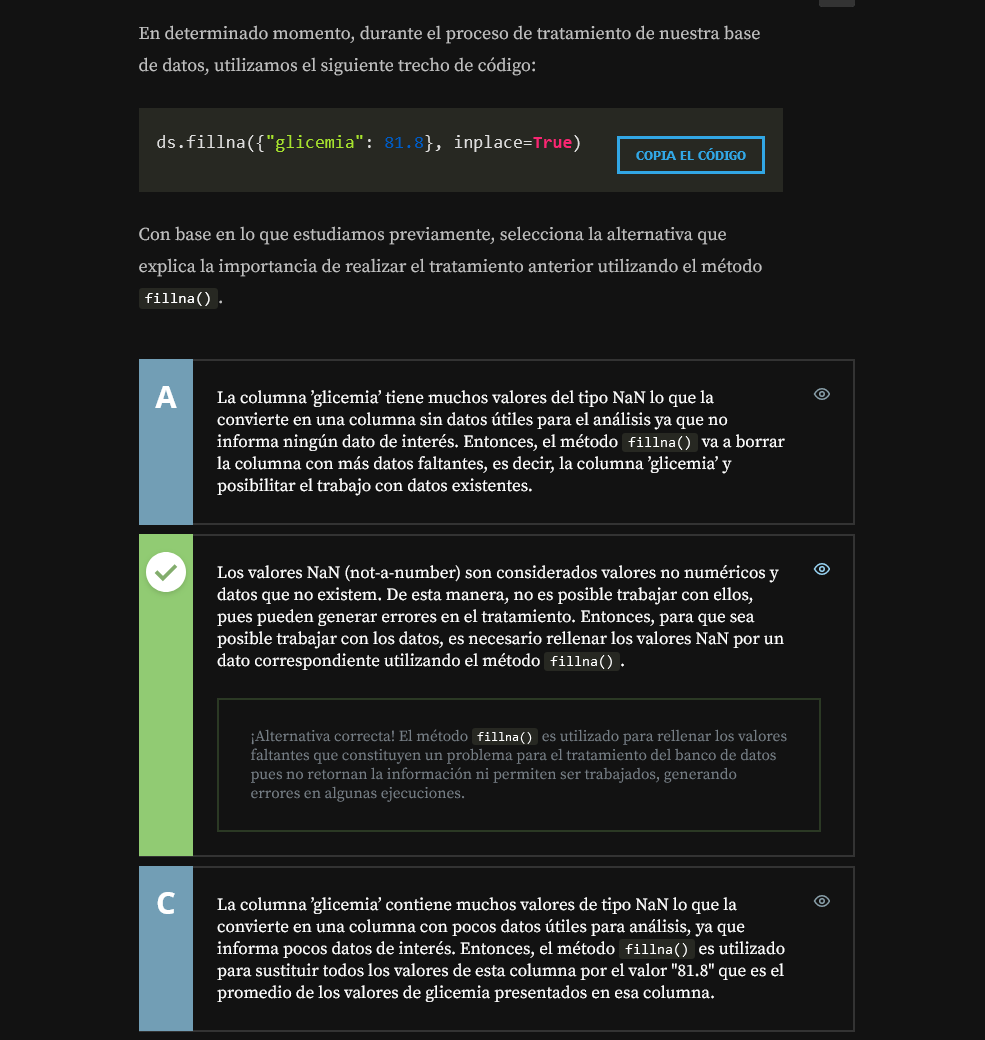
* **ADS**

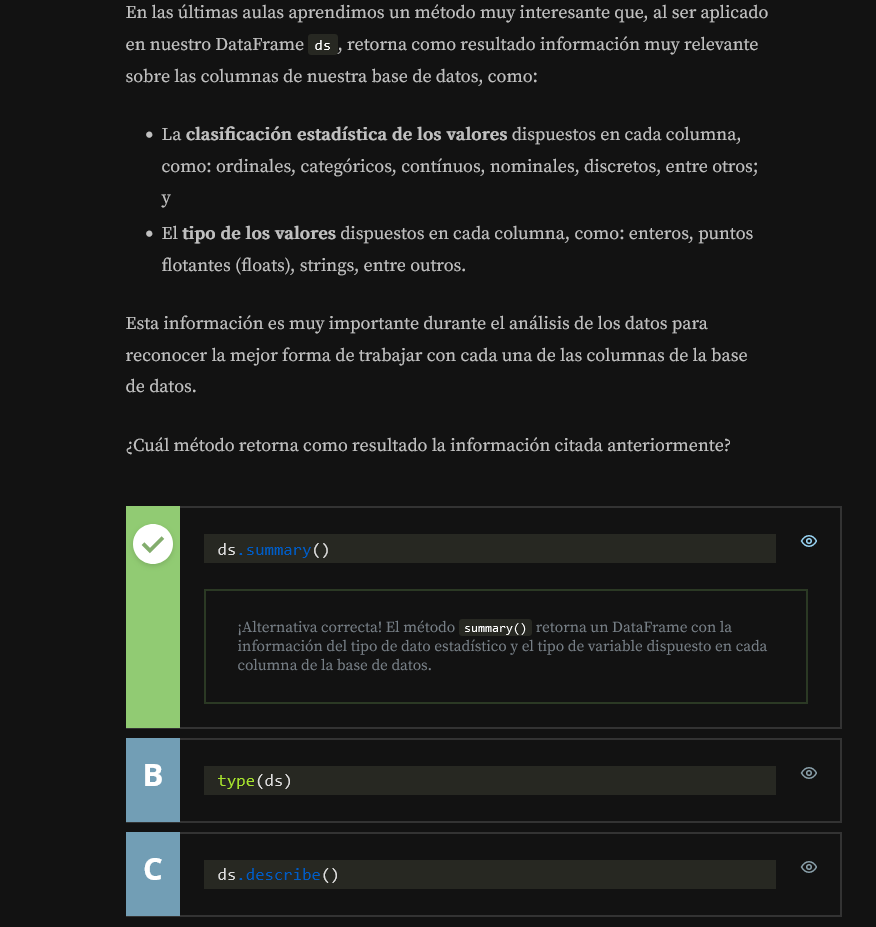
ADS es una biblioteca Python que hace parte del servicio de Data Science de **Oracle Cloud Infrastructure (OCI)**. Esta biblioteca posee una interfaz amigable y métodos que abarcan toda la parte de análisis y de manipulación de datos. El **ADS DataFrame** es una estructura de datos basado en Pandas; de esta forma, cualquier operación que se pueda ejecutar en un **Pandas DataFrame** también puede ser aplicada en un conjunto de datos ADS.

En el ambiente Oracle, es más interesante utilizar funciones específicas de ADS debido a la mayor compatibilidad con el ambiente. Por ello, durante el curso optaremos por utilizar las funciones de ADS en vez de las de otras bibliotecas como Pandas y Dask.









La función cut(), de la biblioteca Pandas, tiene como objetivo separar un conjunto de datos en categorías. Para que esta categorización sea realizada, esta función necesita recibir 2 atributos principales:

* x: matriz con los datos a ser categorizados;
* bins: Lista de límites o cantidade de categorías.

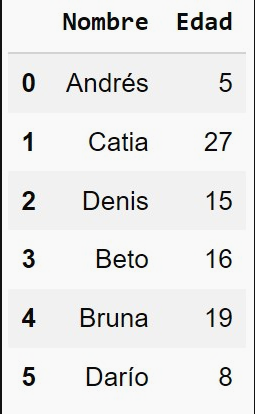
Esta función también posee el atributo labels para el cual podemos pasar los títulos de cada una de las categorías. No obstante, este atributo es opcional, o sea, no necesitamos asignar nombres a cada una de las categorías.

Esta función es muy utilizada cuando queremos categorizar datos contínuos. Para entender mejor la aplicación de esta función, considera el siguiente DataFrame:

df = pd.DataFrame({'Nombre': ['Andrés', 'Catia', 'Denis', 'Beto', 'Bruna', 'Darío'],

'Edad': [5, 27, 15, 16, 19, 8]},

columns = ['Nombre', 'Edad'])



Supongamos que deseamos crear una tercera columna indicando si cada una de esas personas son niños, adolescentes o adultos con base en los valores presentados en la columna "Edad".

Para hacer esto, necesitamos primeramente definir una lista de límites para los valores de la edad, o sea, vamos a definir hasta qué edad la persona debe ser considerada como niño, adolescente e así sucesivamente. De esta manera, vamos a crear la lista limites\_edades:

limites\_edades = [1,13,18,60]

Con esta lista, estamos definiendo 3 intervalos de edad, siendo ellos:

* Intervalo 1: De 1 a 13;
* Intervalo 2: De 14 a 18;
* Intervalo 3: De 19 a 60.

Ahora, vamos a crear una lista con categorías para cada uno de estos intervalos:

categorias = ['Niño(a)', 'Adolescente', 'Adulto(a)']

Con estas dos listas creadas, podemos utilizar la función cut() para generar una tercera columna clasificando a las personas en niño(a), adulto(a) o adolescente de acuerdo con su edad. Para ello, colocaremos como parámetro la columna "Edad" de nuestro DataFrame para el atributo x, la lista limites\_edades para el atributo bins y la lista categorias para el atributo labels:

rango\_edad = pd.cut(x = df.Edad,

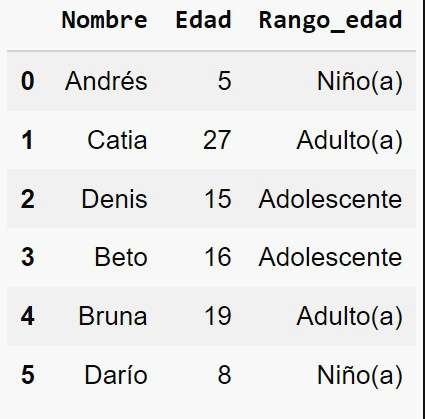
bins = limites\_edades,

labels = categorias)

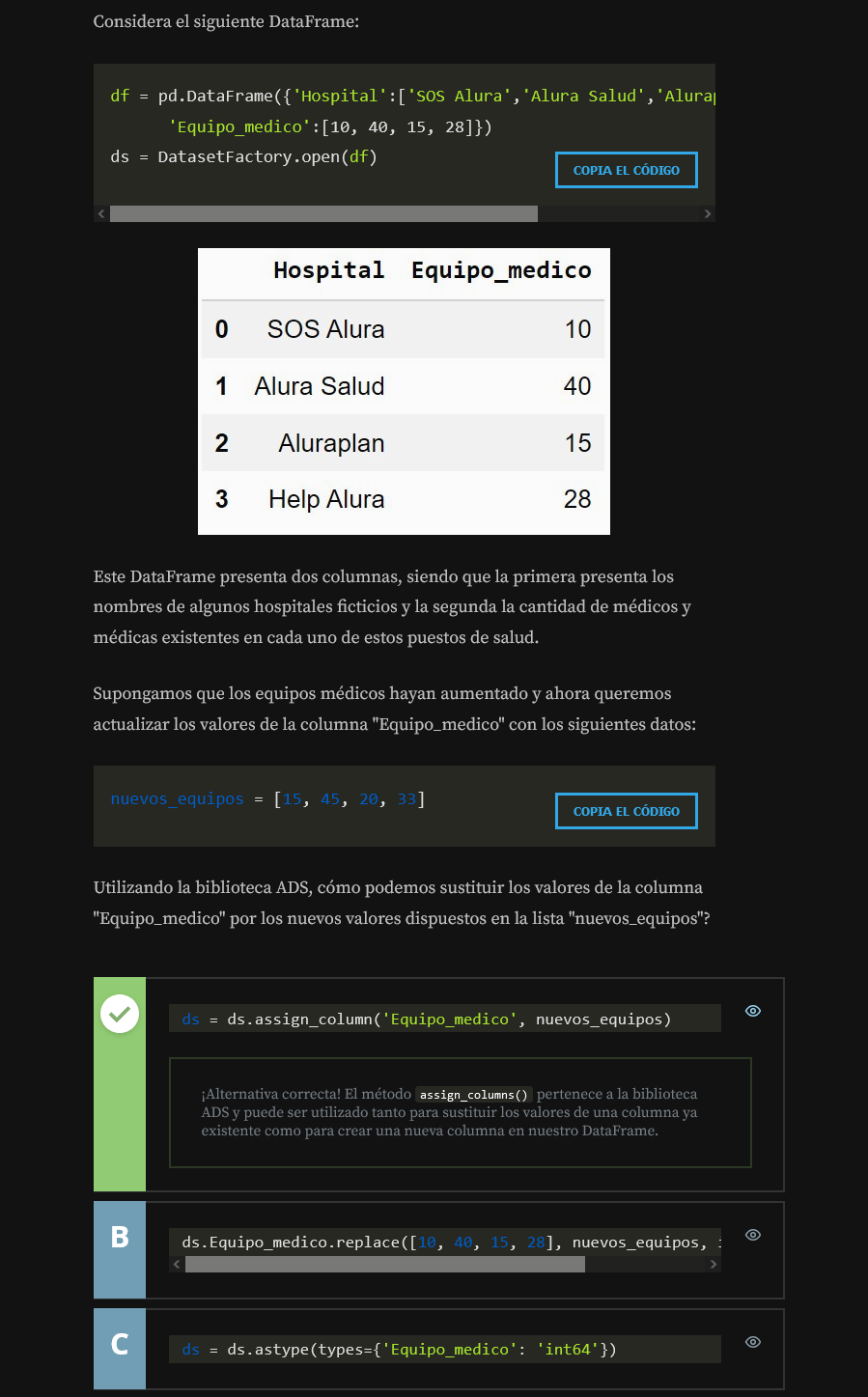
Para finalizar, basta crear una nueva columna en el DataFrame df y configurar la columna rango\_edad con el resultado de la función cut():

df['Rango\_edad'] = rango\_edad

El DataFrame resultante será igual a:



¡Te felicito! Ahora nuestro DataFrame df posee también la columna "Rango\_edad" con las categorías que definimos de acuerdo con la columna "Edad".





La correlación **determina el grado de asociación entre dos variables**, por ejemplo:

* El precio de un carro y el precio de su seguro;
* la edad y el costo de un plan de salud;
* Masa corporal y altura.

Pero no siempre esta asociación entre las variables es una relación fácil de ser identificada. De esa forma, existen los coeficientes de correlación que son los responsables por determinar el grado de "fuerza" de la correlación entre dos variables y también su dirección.

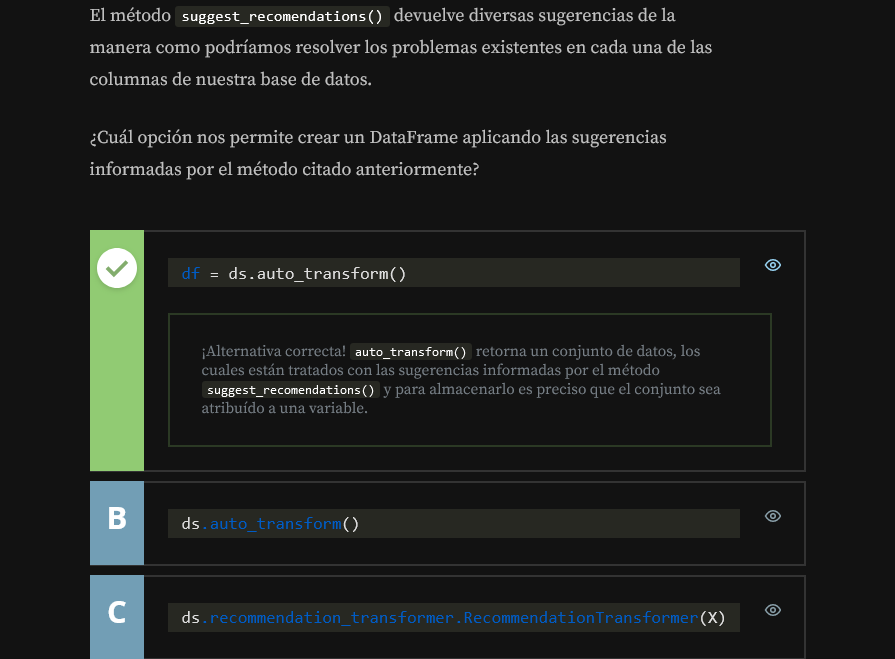
**Coeficiente de Correlación de Pearson**

También conocido como "ρ de Pearson" mide cuánto dos variables contínuas son correlacionadas. Dicho coeficiente puede variar de -1 a 1, donde estos valores indican:

* Valores de 0 a -0.3 ou 0 a 0.3: correlación irrelevante;
* Valores de -0.3 a -0.5 ou 0.3 a 0.5: correlación debil;
* Valores de -0.5 a -0.7 ou 0.5 a 0.7: correlación moderada;
* Valores de -0.7 a -0.9 ou 0.7 a 0.9: correlación fuerte;
* Valores de -0.9 a -1 ou 0.9 a 1: correlação muy fuerte.

Para medir el grado de correlación de variables que no son contínuas existen otros coeficientes de correlación que pueden ser utilizados, como el de Spearman, Kendall, entre otros. Si deseas conocer un poco más sobre estos otros métodos, te sugiero leer más sobre [Correlación](https://www.probabilidadyestadistica.net/correlacion/" \t "https://app.aluracursos.com/course/oracle-ads-analisis-datos-nube/task/_blank).





Los datos desbalanceados son aquellos que poseen muchos registros para una categoría y pocos para otra. Por ejemplo, imagina que estamos trabajando con datos sobre el oceano y queremos verificar las condiciones en las cuales se presenta un maremoto.

Si tenemos datos de todos los días de un año, es bastante probable que existan muchos más datos de mar normal que de maremoto. Con eso, podemos tener dificultades en localizar los datos que se refieren a esta condición específica.

De la misma forma, si nuestro interés es crear algún modelo que pueda prever, basado en los datos, si habrá un tsunami, el modelo puede tener dificultades debido a la pequeña cantidad de datos sobre maremoto.

Por ello, existen técnicas conocidas como balanceo de datos. Estas técnicas realizan precisamente un balanceo, igualando el número de registros que se refieren a días con y sin maremoto.

Estas técnicas se dividen en dos vertientes principales: upsampling o sobremuestreo y downsamping o submuestreo.

Las técnicas de sobremuestreo consisten en crear (basados en los registros existentes) nuevos registros para la clase que aparece menos, hasta que la cantidad de registros para ambas clases se iguale. Por otro lado, las técnicas de submuestreo consisten en excluir registros de la clase que más aparece hasta que se tenga la misma cantidad de registros para ambas clases.

Cada una de estas técnicas se adecúa mejor a diversas situaciones. A continuación discutiremos algunas ventajas y desventajas de cada una de estas técnicas.

**Upsampling**

Ventajas

* + Aumenta el número de registros.
  + Facilita a criação de modelos de ML.

Desventajas

* + Crea distorsiones al Dataframe.
  + Genera registros que no corresponden con la realidad.

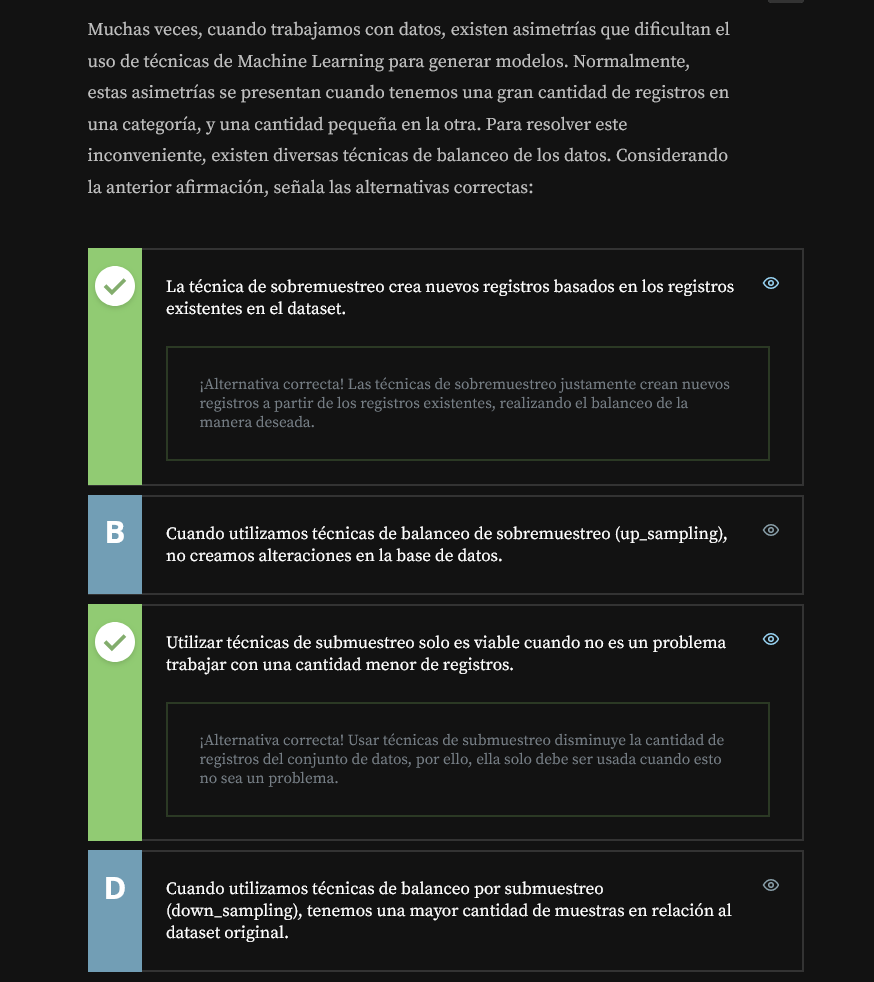
**Downsampling**

Ventajas

* + No cria distorções no Dataframe.

Desventajas

* + Disminuye la cantidad de registros.



# Machine Leaning con Oracle ADS