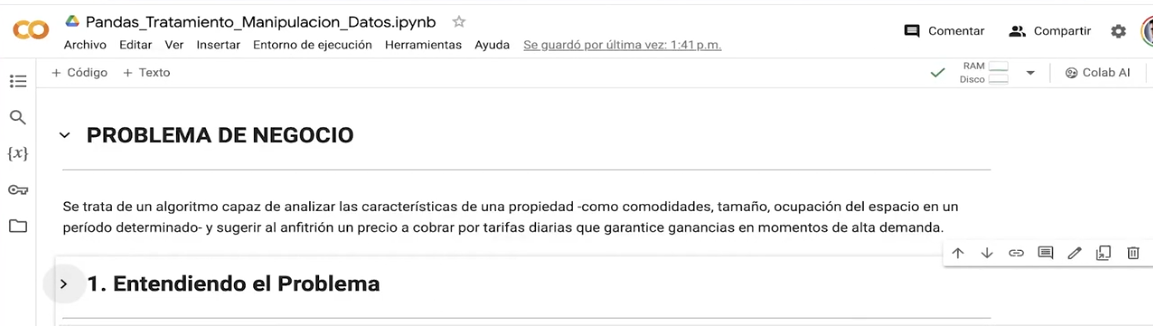
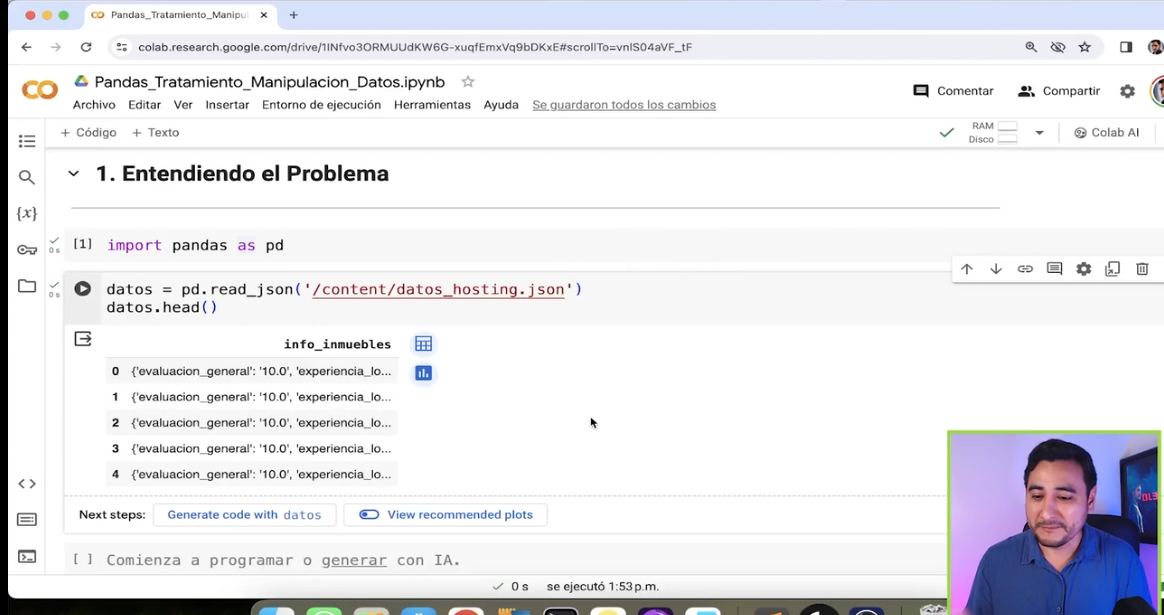
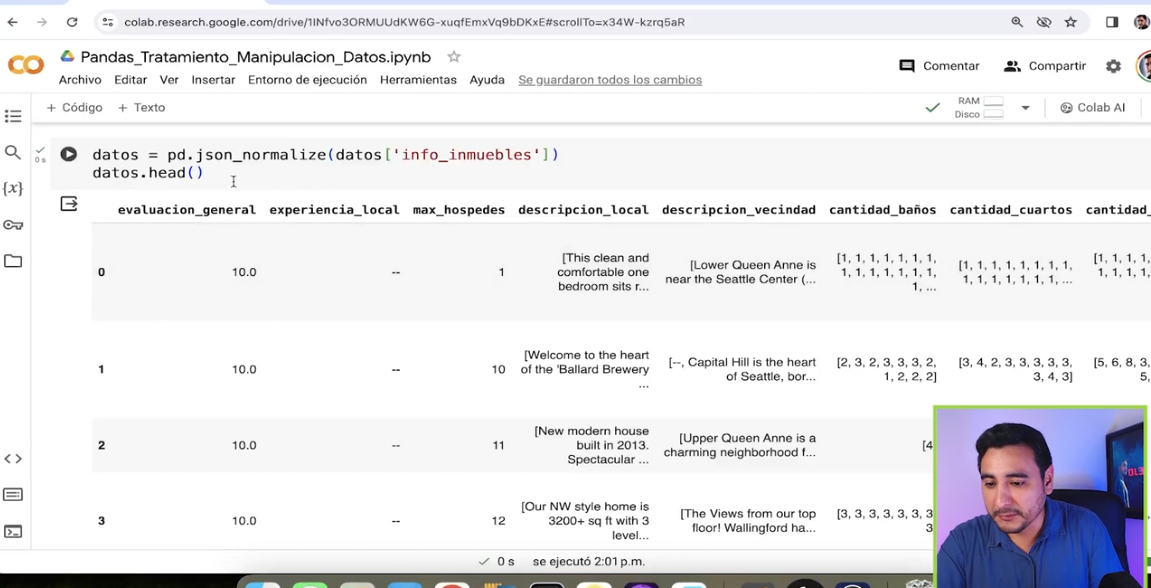
**Transformación y Manipulación de Datos**

**Conociendo el problema**

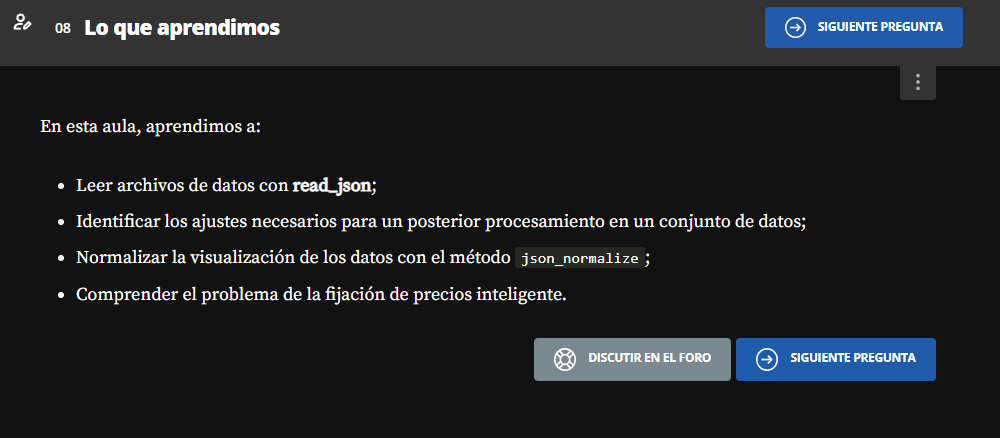
****

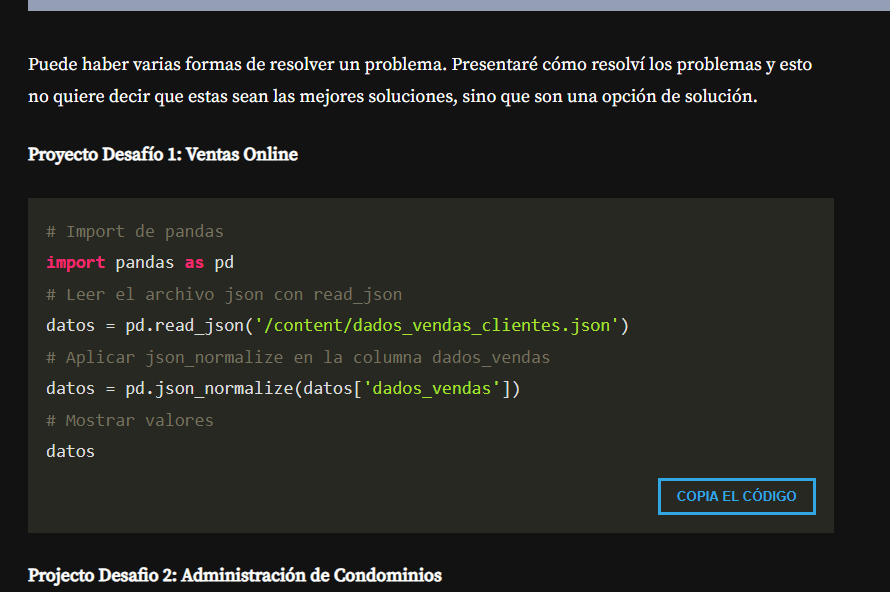
**** ****

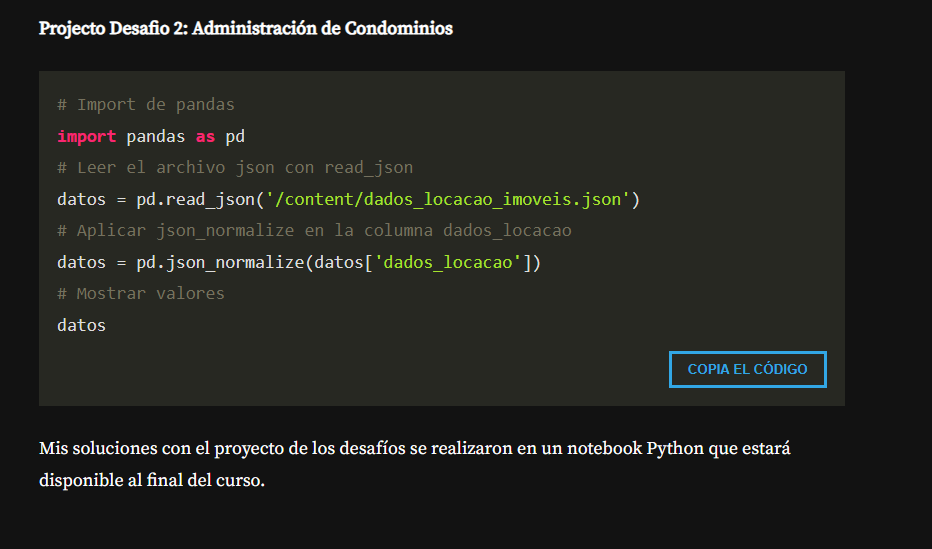
****

**Se aplica el normalize para desanidar una columna con diccionarios**

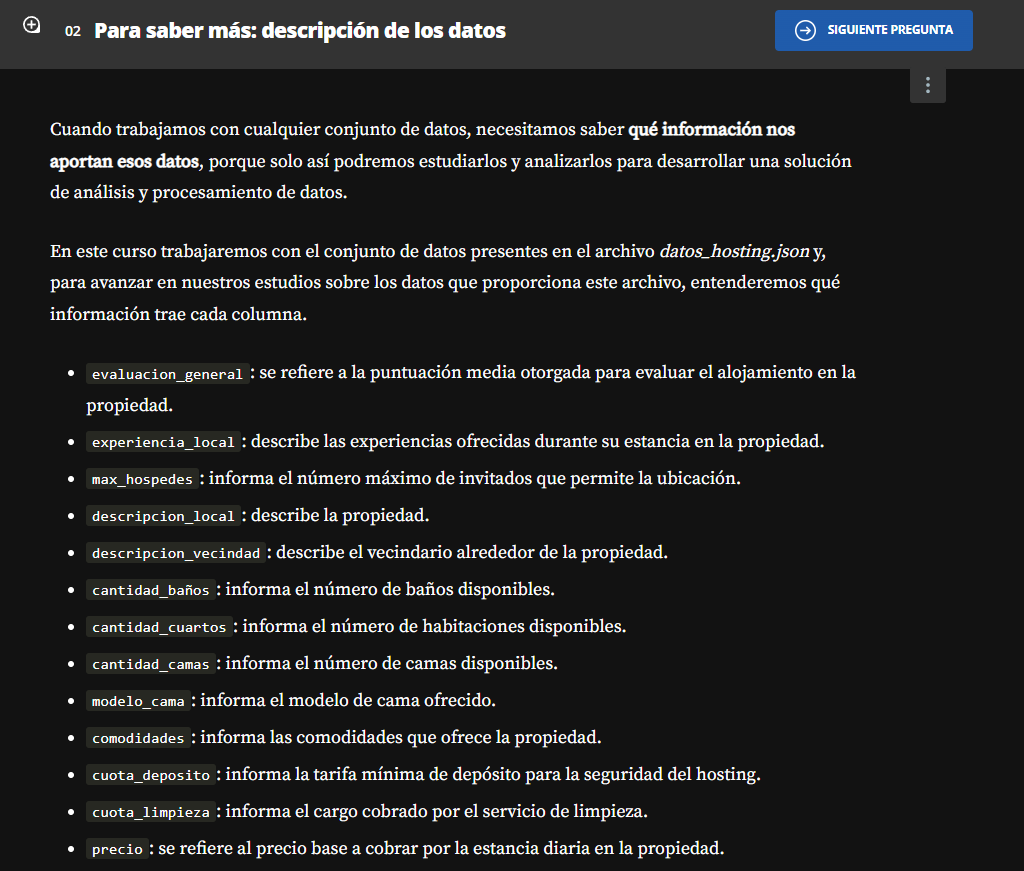
****

****

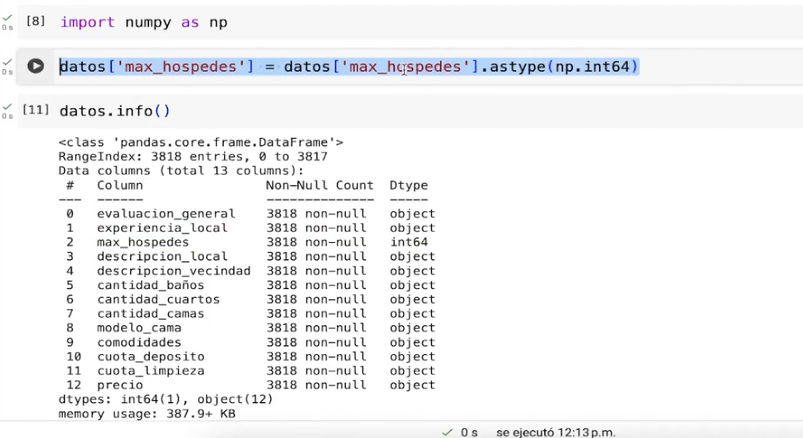
****

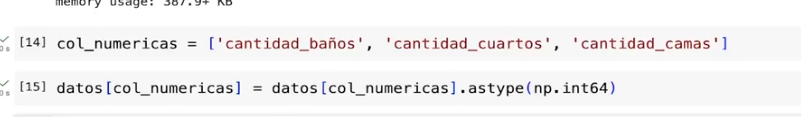
****

**Datos Númericos**

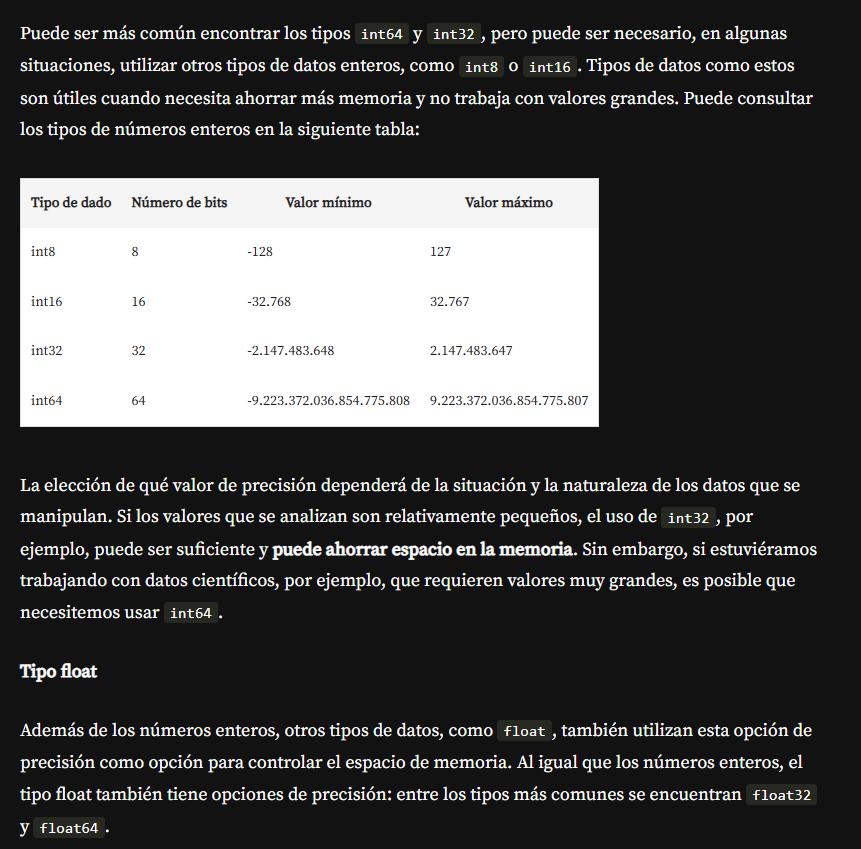
****

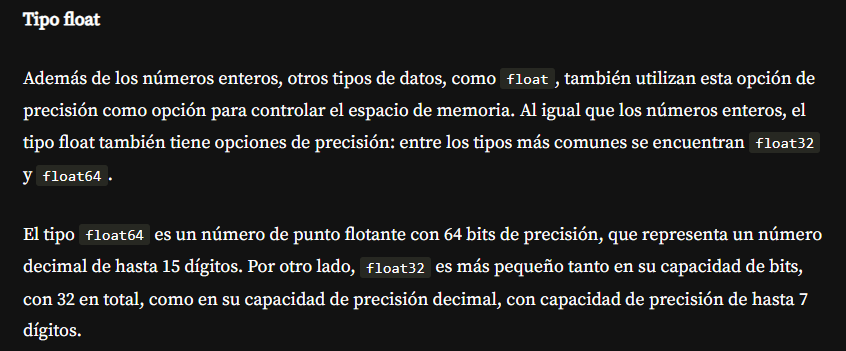
**Convirtiendo objetos a datos númericos y guardandolos en el dataframe**

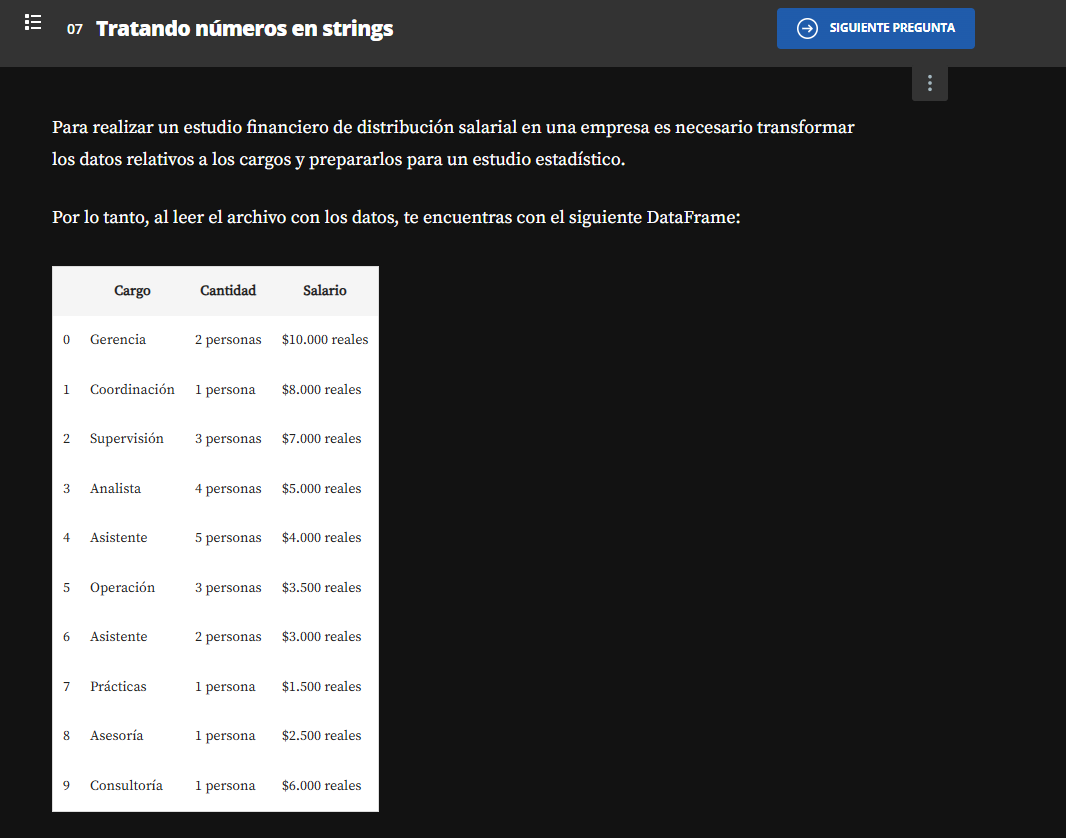
****

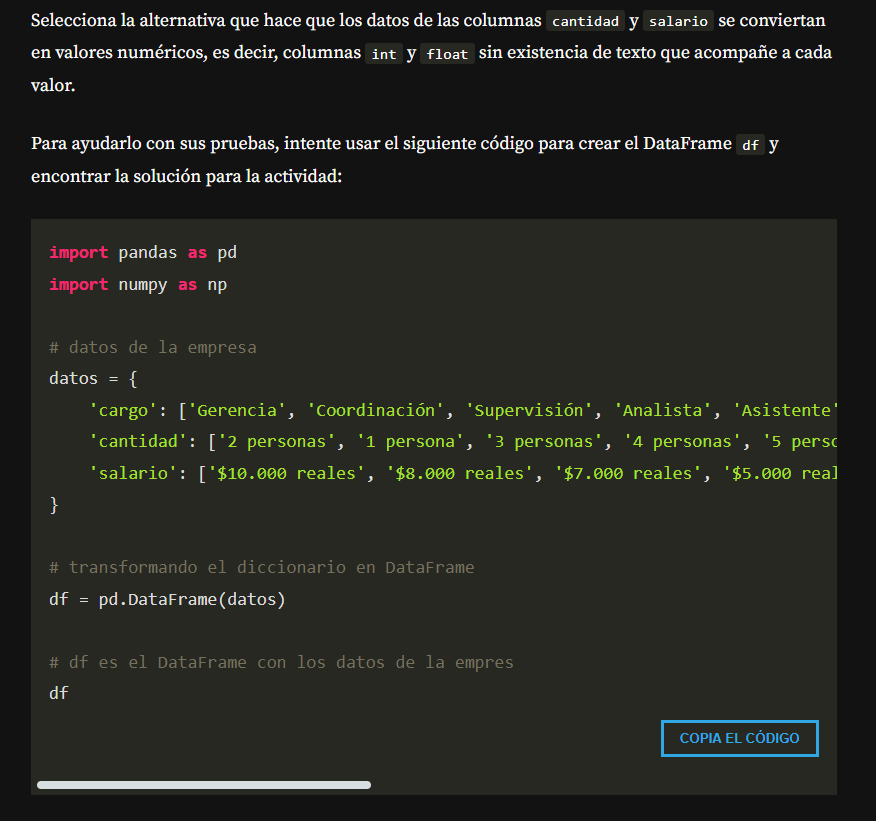
****

****

****

****

****

****

**import** pandas **as** pd

**import** numpy **as** np

# datos de la empresa

datos = {

'cargo': ['Gerencia', 'Coordinación', 'Supervisión', 'Analista', 'Asistente', 'Operación', 'Asistente', 'Prácticas', 'Asesoría', 'Consultoría'],

'cantidad': ['2 personas', '1 persona', '3 personas', '4 personas', '5 personas', '3 personas', '2 personas', '1 persona', '1 persona', '1 persona'],

'salario': ['$10.000 reales', '$8.000 reales', '$7.000 reales', '$5.000 reales', '$4.000 reales', '$3.500 reales', '$3.000 reales', '$1.500 reales', '$2.500 reales', '$6.000 reales']

}

# transformando el diccionario en DataFrame

df = pd.DataFrame(datos)

# df es el DataFrame con los datos de la empres

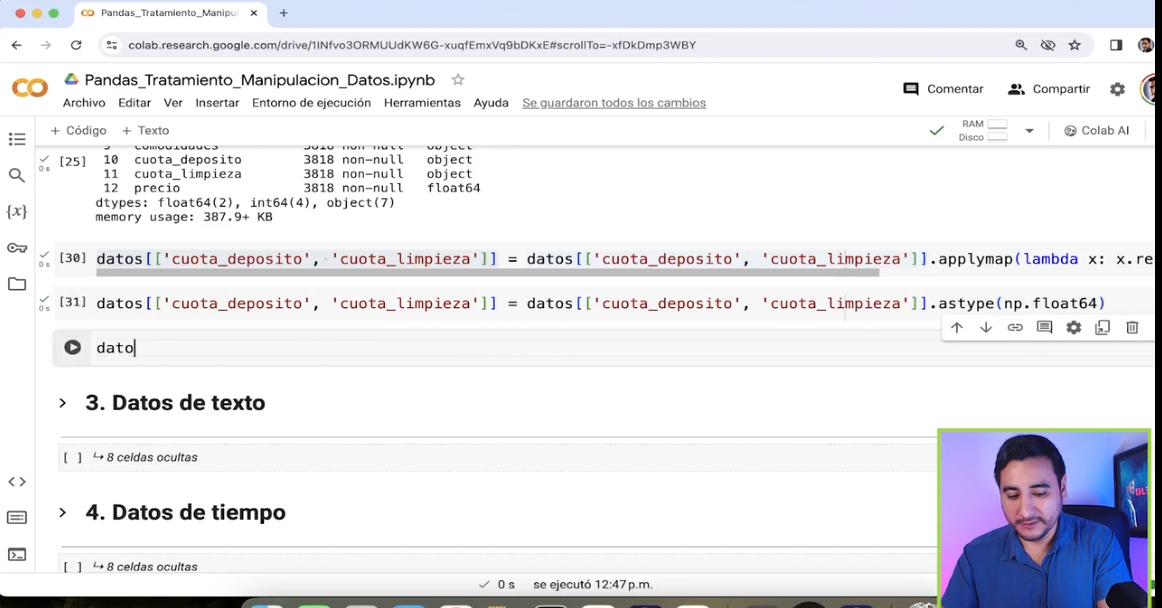
df

****

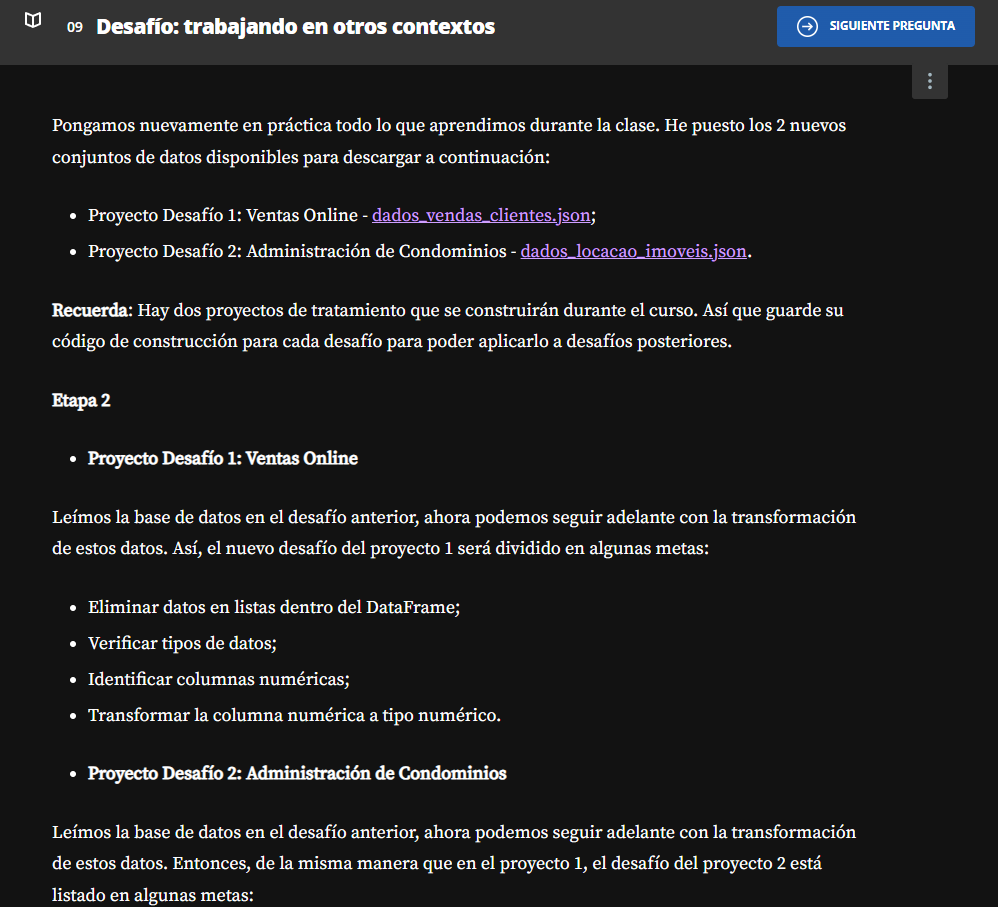
**Esto es para convertir multiples columnas de un dataframe**

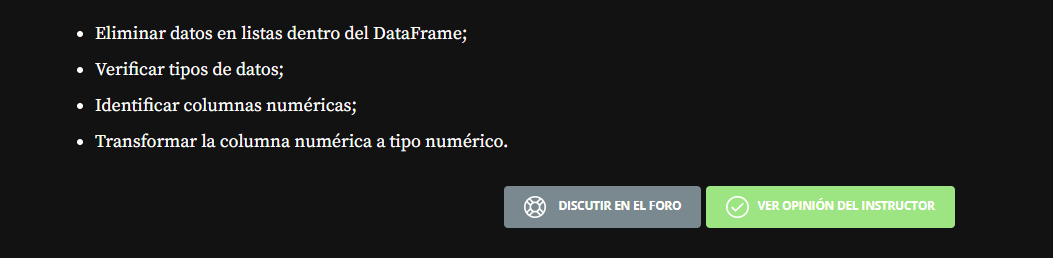
****

**Luego se transforman los datos**

****

**Hagamos lo que hicimos**

****

****

# Colectar los valores de las columnas y verificar

columnas = list(datos.columns)

columnas

# Destrincar las listas con explode

datos = datos.explode(columnas[1:])

# Resetear los index de las líneas

datos.reset\_index(drop=**True**,inplace=**True**)

# Observar el DataFrame

datos

# Verificar los tipos de datos con info

datos.info()

# La columna numérica es el 'Valor da compra'

datos['Valor da compra']

# Iniciar la transformación

# Import de la biblioteca numpy

**import** numpy **as** np

# Remover los textos presentes en la base

# Cambiar las comas separadoras del decimal por punto

datos['Valor da compra'] = datos['Valor da compra'].apply(**lambda** x: x.replace('R$ ', '').replace(',','.').strip())

# Cambiar los tipo de datos para float

datos['Valor da compra'] = datos['Valor da compra'].astype(np.float64)

# Verificar la transformación

datos.info()

****

# Colectar los valores de las columnas y verificar

columnas = list(datos.columns)

columnas

# Destrincar las listas con explode

datos = datos.explode(columnas[1:])

# Resetear los index de las líneas

datos.reset\_index(drop=**True**,inplace=**True**)

# Observar el DataFrame

datos

# Verificar los tipos de datos con info

datos.info()

# La columna numérica es el 'valor\_aluguel'

datos['valor\_aluguel']

# Iniciar la transformación

# Import de la biblioteca numpy

**import** numpy **as** np

# Remover los textos presentes en la base

# Cambiar las comas separadoras del decimal por punto

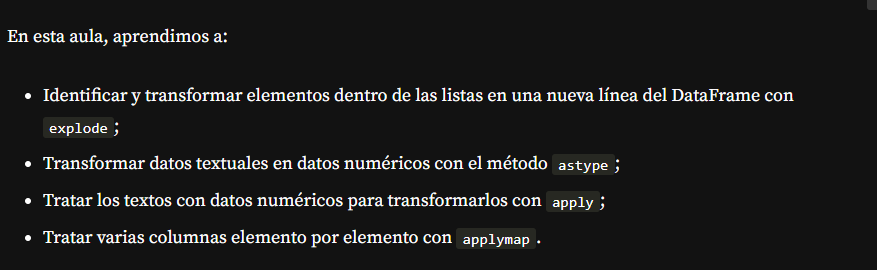
datos['valor\_aluguel'] = datos['valor\_aluguel'].apply(**lambda** x: x.replace('$ ', '').replace(' reais', '').replace(',','.').strip())

# Cambiar los tipos de datos para float

datos['valor\_aluguel'] = datos['valor\_aluguel'].astype(np.float64)

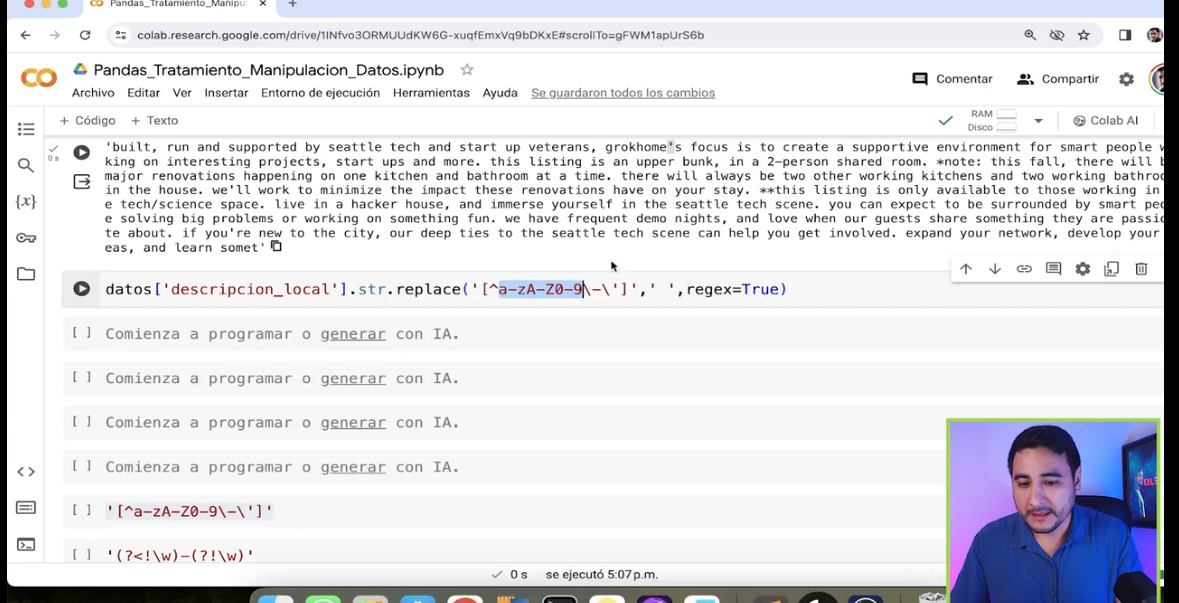
# Verificar la transformación

datos.info()

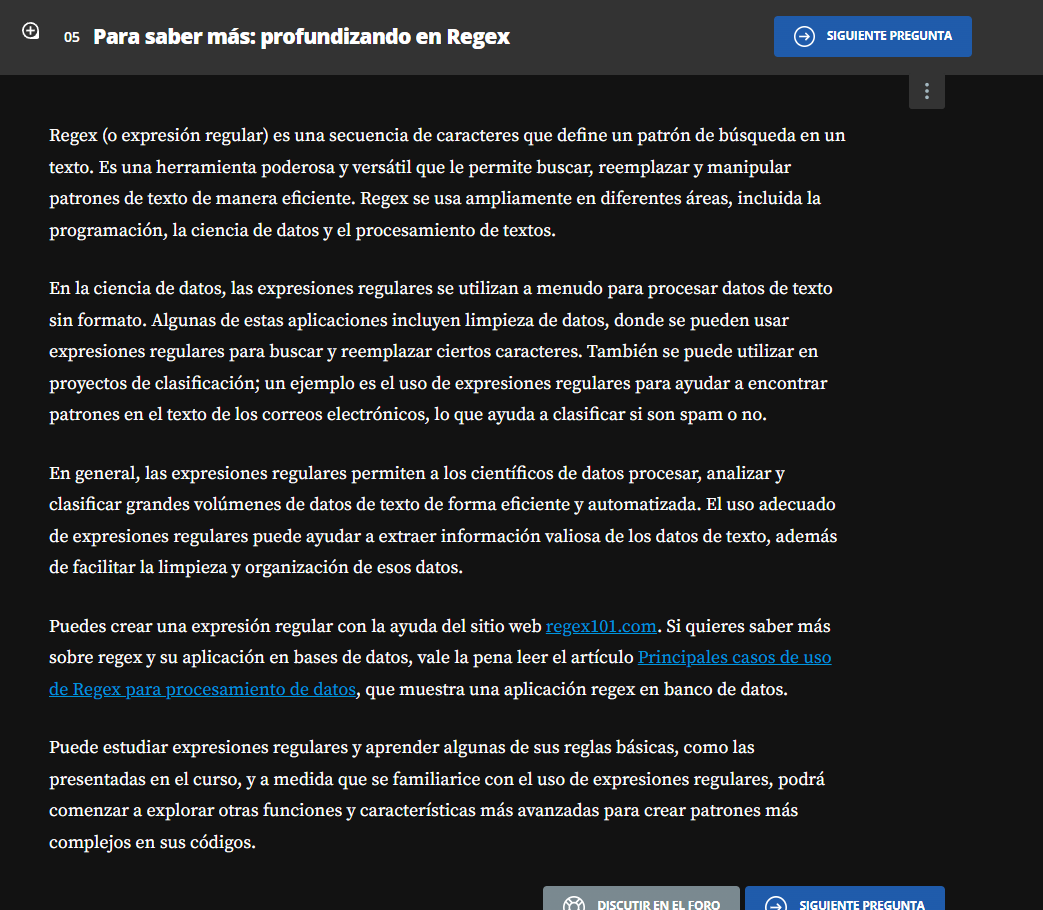
****

**Datos de Texto**

****

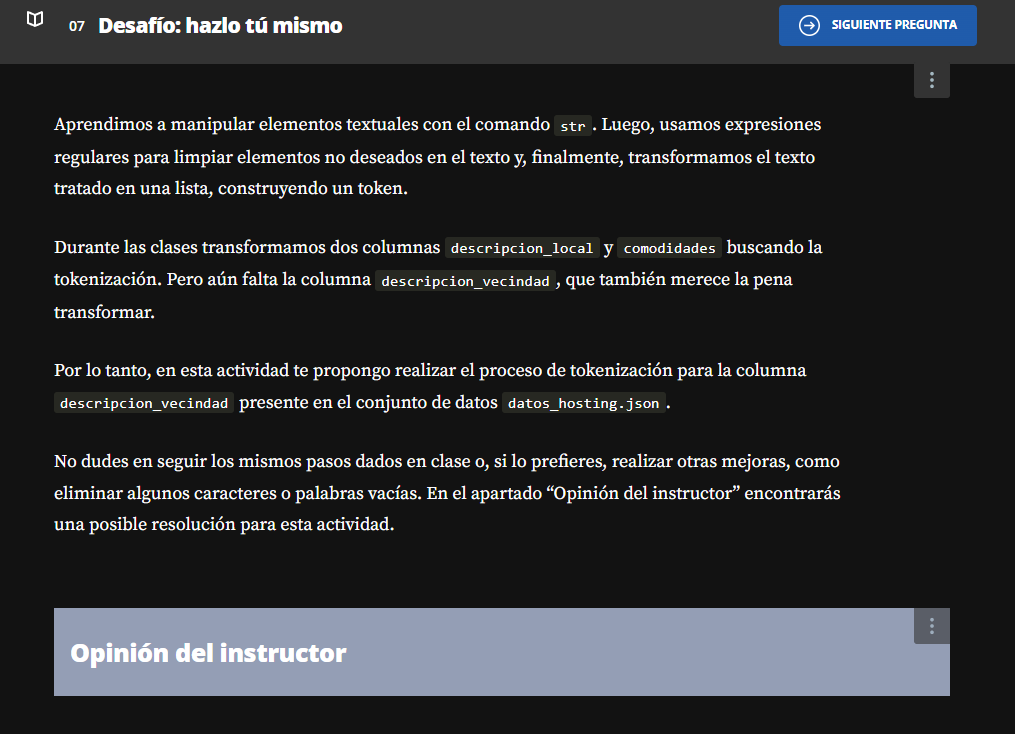
****

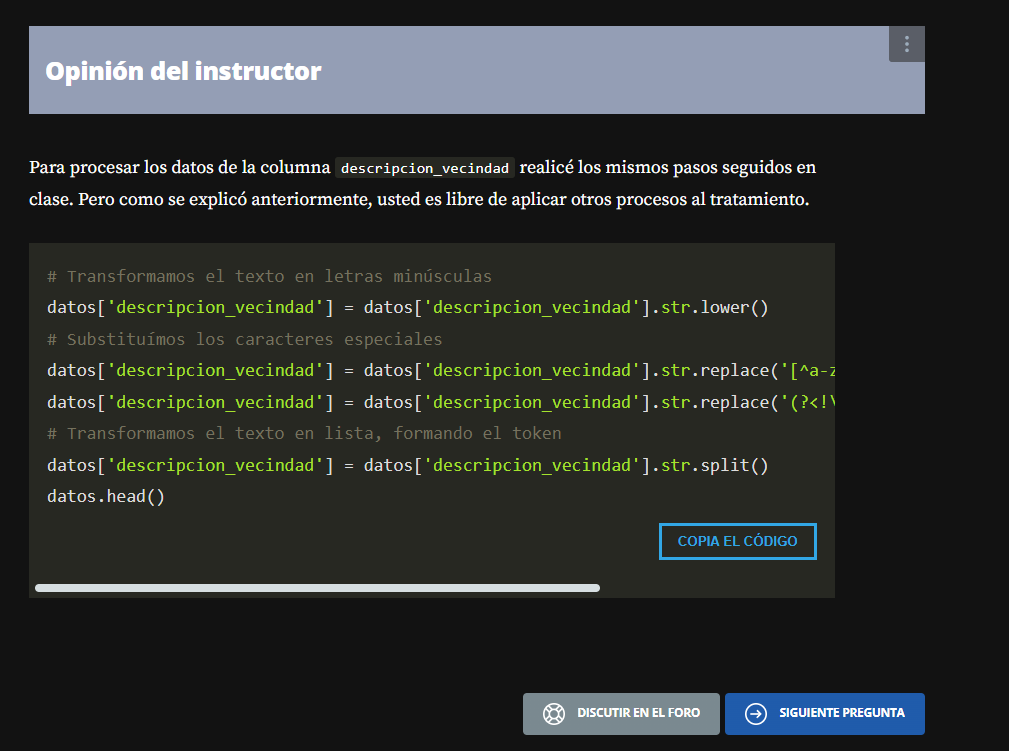
**Estructuras de negación para remplazar en el texto**

****

[**https://regex101.com/**](https://regex101.com/)

[**https://www.alura.com.br/artigos/principais-casos-uso-regex-para-tratamento-dados?\_gl=1\*16oatds\*\_gcl\_au\*MTA5MjAwNjMxNi4xNzQ2MjQ5MDg0\*\_ga\*MjM5OTE1MTQzLjE3MzY2MDY5MTM.\*\_ga\_WWRP4FFDZK\*czE3NTMxNDkxNDQkbzk1JGcxJHQxNzUzMTU4NzM4JGo2MCRsMCRoMA**](https://www.alura.com.br/artigos/principais-casos-uso-regex-para-tratamento-dados?_gl=1*16oatds*_gcl_au*MTA5MjAwNjMxNi4xNzQ2MjQ5MDg0*_ga*MjM5OTE1MTQzLjE3MzY2MDY5MTM.*_ga_WWRP4FFDZK*czE3NTMxNDkxNDQkbzk1JGcxJHQxNzUzMTU4NzM4JGo2MCRsMCRoMA)**..**

****

****

# Transformamos el texto en letras minúsculas

datos['descripcion\_vecindad'] = datos['descripcion\_vecindad'].str.lower()

# Substituímos los caracteres especiales

datos['descripcion\_vecindad'] = datos['descripcion\_vecindad'].str.replace('[^a-zA-Z0-9\-\']', ' ', regex=**True**)

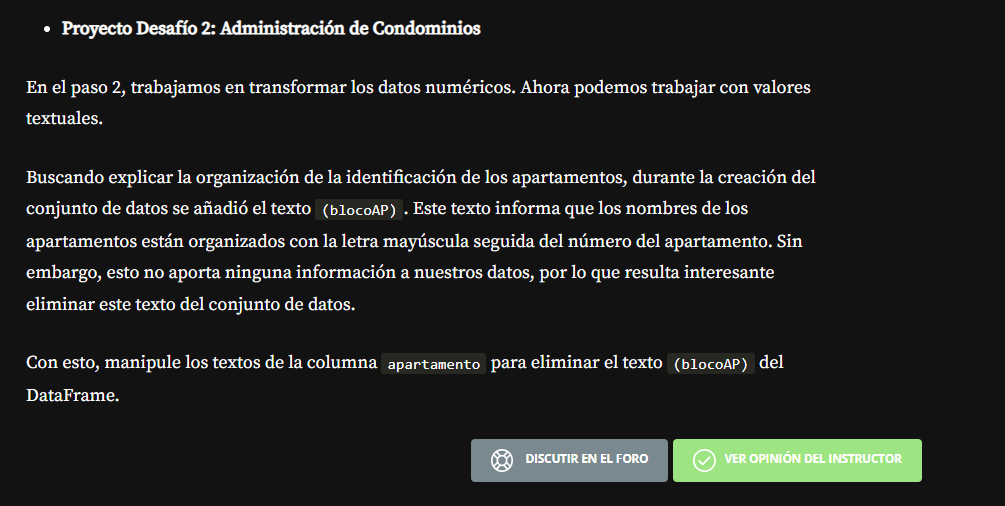
datos['descripcion\_vecindad'] = datos['descripcion\_vecindad'].str.replace('(?<!\w)-(?!\w)', '', regex=**True**)

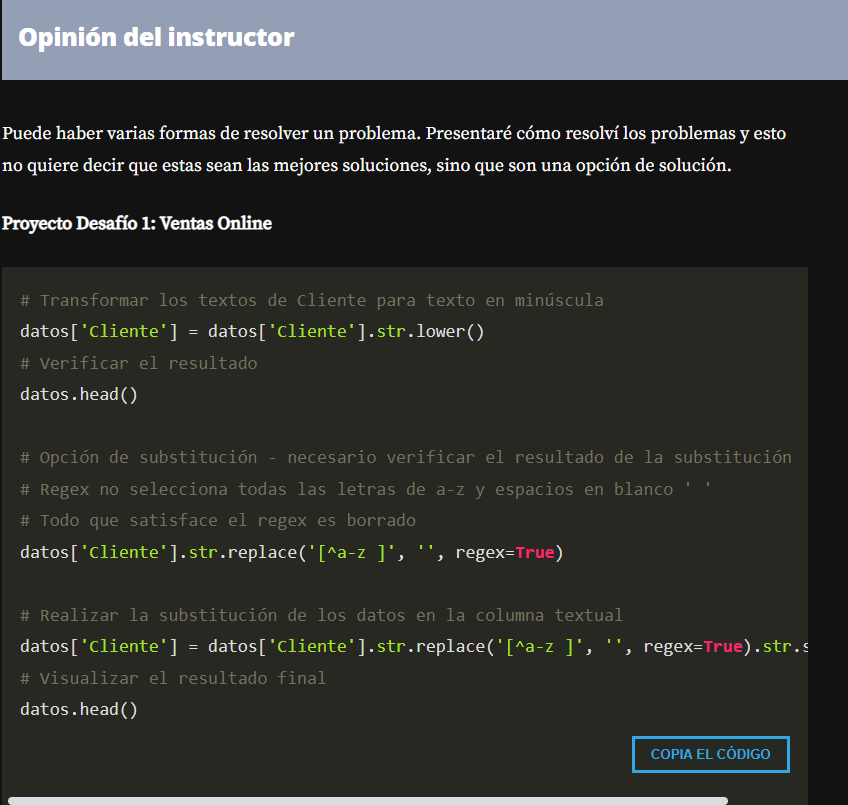
# Transformamos el texto en lista, formando el token

datos['descripcion\_vecindad'] = datos['descripcion\_vecindad'].str.split()

datos.head()



****

****

# Transformar los textos de Cliente para texto en minúscula

datos['Cliente'] = datos['Cliente'].str.lower()

# Verificar el resultado

datos.head()

# Opción de substitución - necesario verificar el resultado de la substitución

# Regex no selecciona todas las letras de a-z y espacios en blanco ' '

# Todo que satisface el regex es borrado

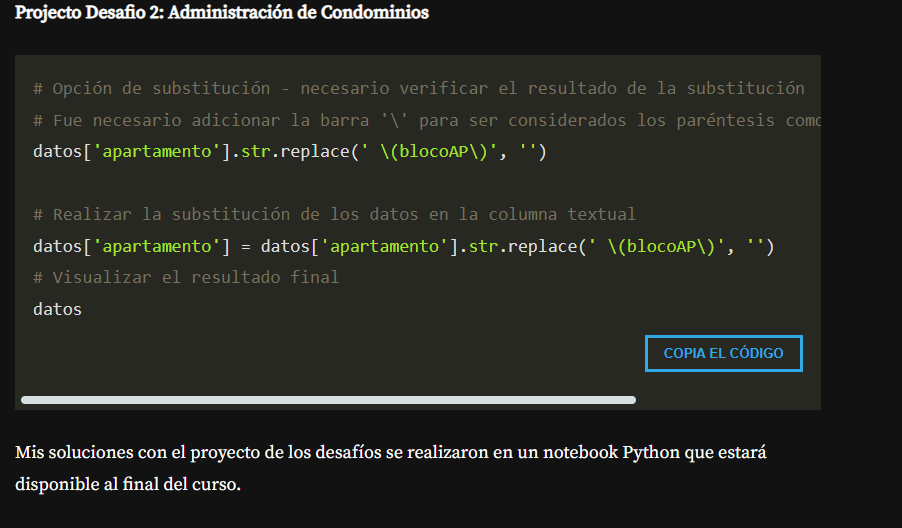
datos['Cliente'].str.replace('[^a-z ]', '', regex=**True**)

# Realizar la substitución de los datos en la columna textual

datos['Cliente'] = datos['Cliente'].str.replace('[^a-z ]', '', regex=**True**).str.strip()

# Visualizar el resultado final

datos.head()

****

# Opción de substitución - necesario verificar el resultado de la substitución

# Fue necesario adicionar la barra '\' para ser considerados los paréntesis como caracteres literales

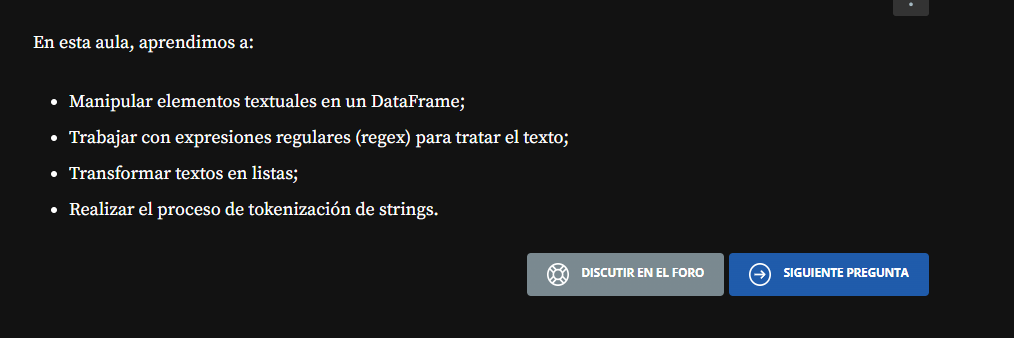
datos['apartamento'].str.replace(' \(blocoAP\)', '')

# Realizar la substitución de los datos en la columna textual

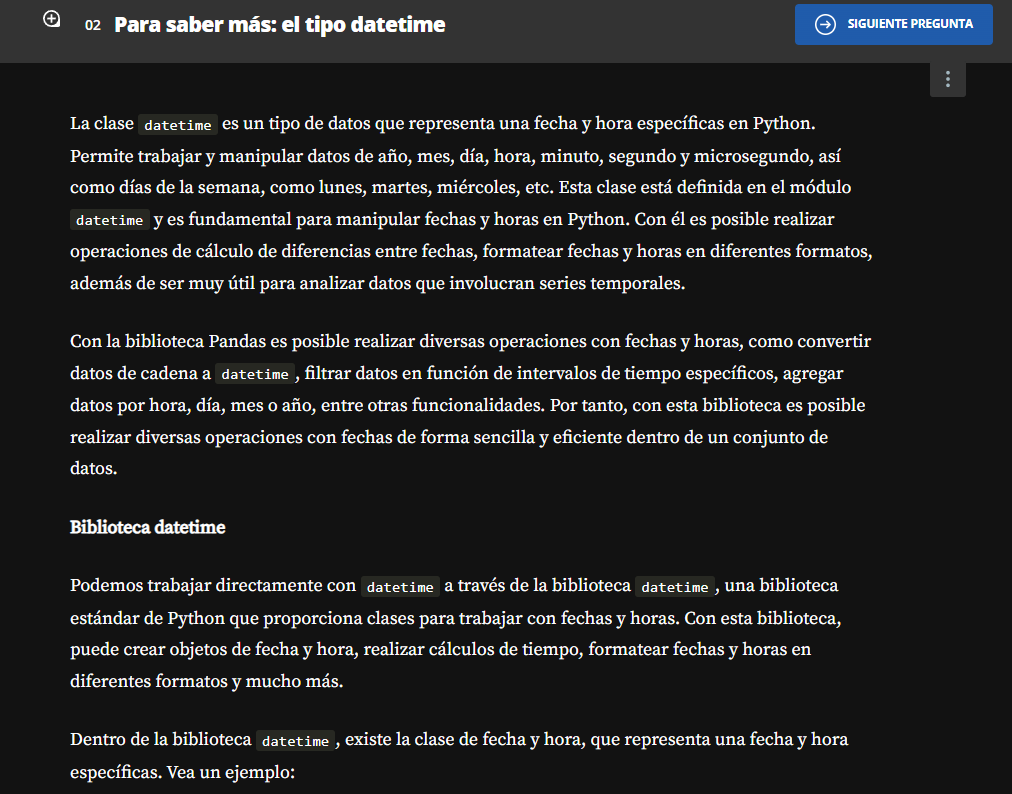
datos['apartamento'] = datos['apartamento'].str.replace(' \(blocoAP\)', '')

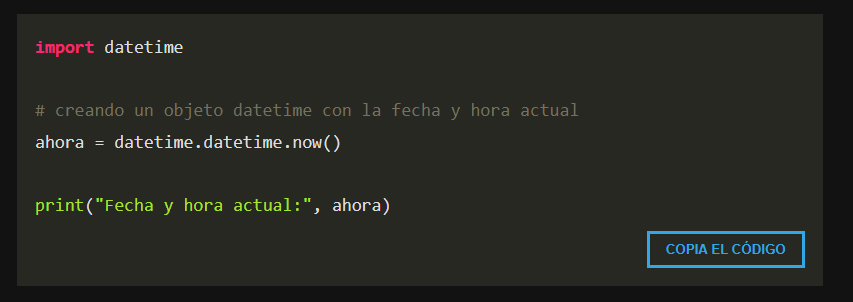
# Visualizar el resultado final

datos

****

**DATOS DE TIEMPO**

****

****

**import** datetime

# creando un objeto datetime con la fecha y hora actual

ahora = datetime.datetime.now()

print("Fecha y hora actual:", ahora)

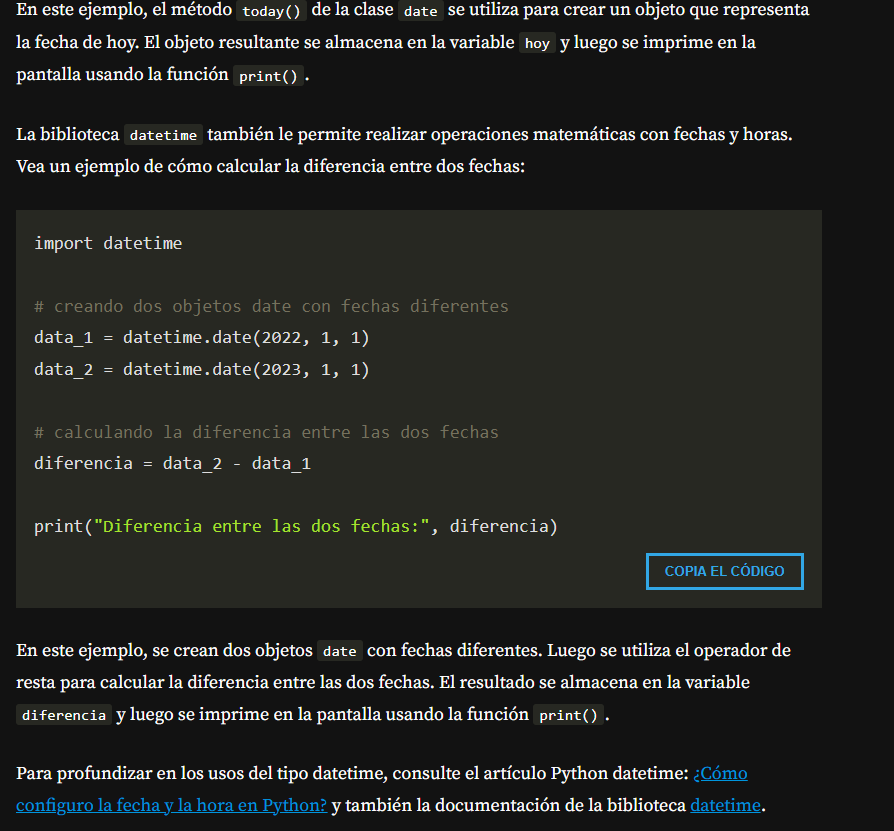
****

**import** datetime

# creando un objeto date con la fecha de hoy

hoy = datetime.date.today()

print("Fecha de hoy:", hoy)

****

import datetime

# creando dos objetos date con fechas diferentes

data\_1 = datetime.date(2022, 1, 1)

data\_2 = datetime.date(2023, 1, 1)

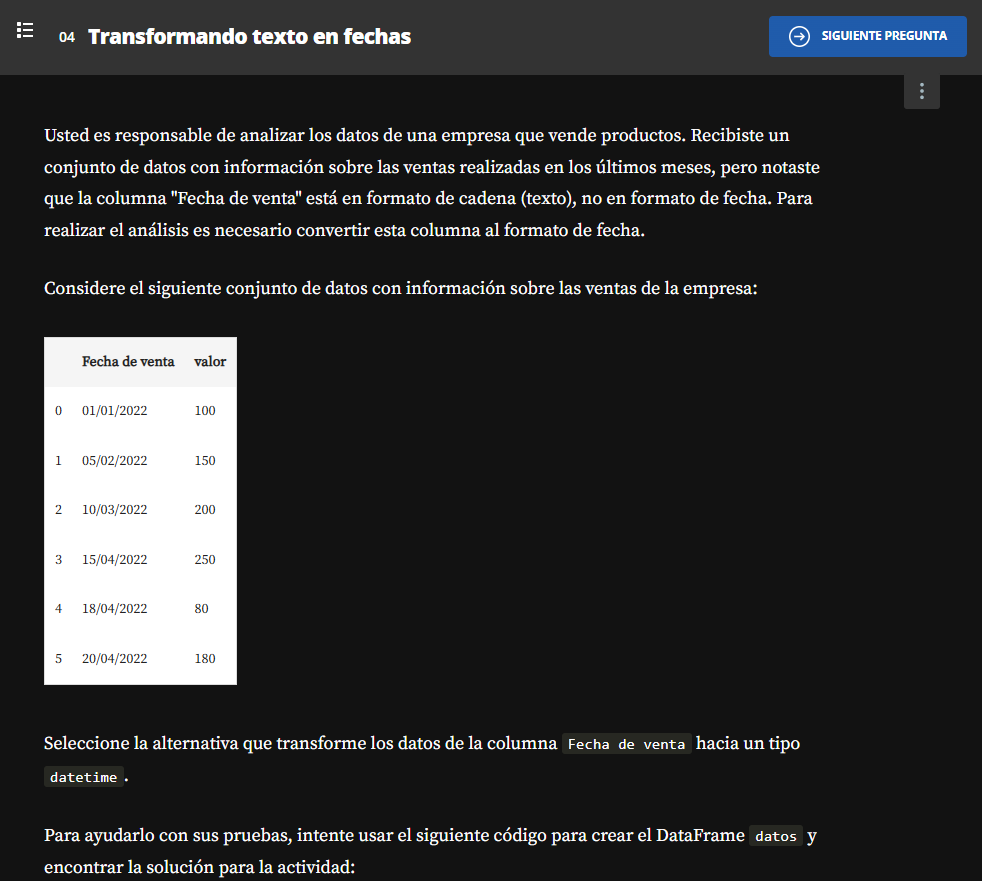
# calculando la diferencia entre las dos fechas

diferencia = data\_2 - data\_1

print("Diferencia entre las dos fechas:", diferencia)

[**https://www.alura.com.br/artigos/lidando-com-datas-e-horarios-no-python?\_gl=1\*d95fpc\*\_gcl\_au\*MTA5MjAwNjMxNi4xNzQ2MjQ5MDg0\*\_ga\*MjM5OTE1MTQzLjE3MzY2MDY5MTM.\*\_ga\_WWRP4FFDZK\*czE3NTMxNDkxNDQkbzk1JGcxJHQxNzUzMTYwMDM3JGo2MCRsMCRoMA**](https://www.alura.com.br/artigos/lidando-com-datas-e-horarios-no-python?_gl=1*d95fpc*_gcl_au*MTA5MjAwNjMxNi4xNzQ2MjQ5MDg0*_ga*MjM5OTE1MTQzLjE3MzY2MDY5MTM.*_ga_WWRP4FFDZK*czE3NTMxNDkxNDQkbzk1JGcxJHQxNzUzMTYwMDM3JGo2MCRsMCRoMA)**..**

[**https://docs.python.org/3/library/datetime.html**](https://docs.python.org/3/library/datetime.html)

****

**import** pandas **as** pd

# Creando el DataFrame con las informaciones

datos = pd.DataFrame({

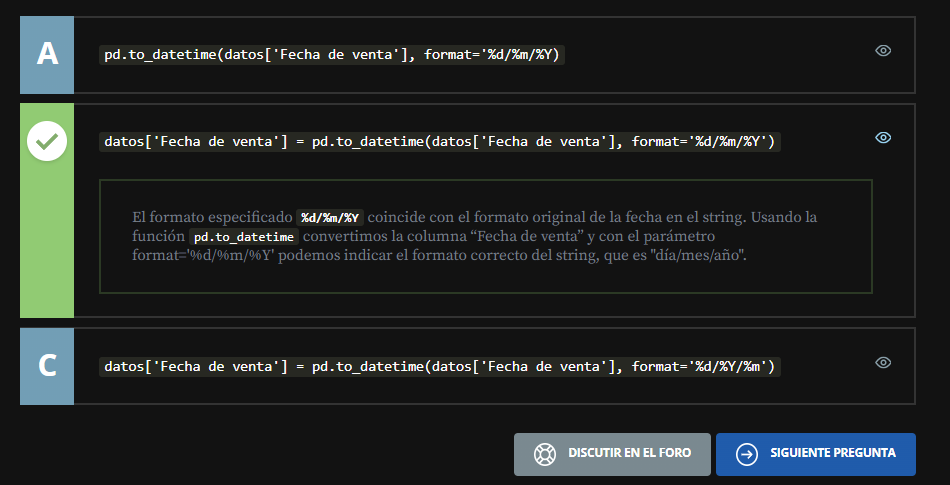
'Fecha de venta': ['01/01/2022', '05/02/2022', '10/03/2022', '15/04/2022','18/04/2022','20/04/2022'],

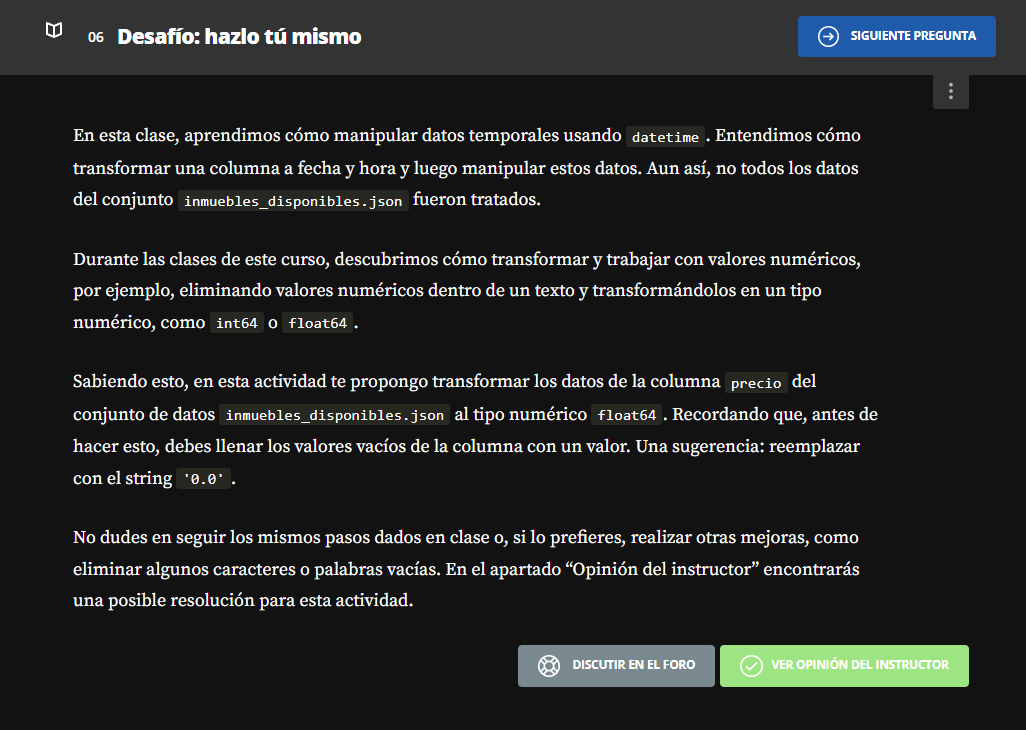
'valor': [100, 150, 200, 250,80,180]

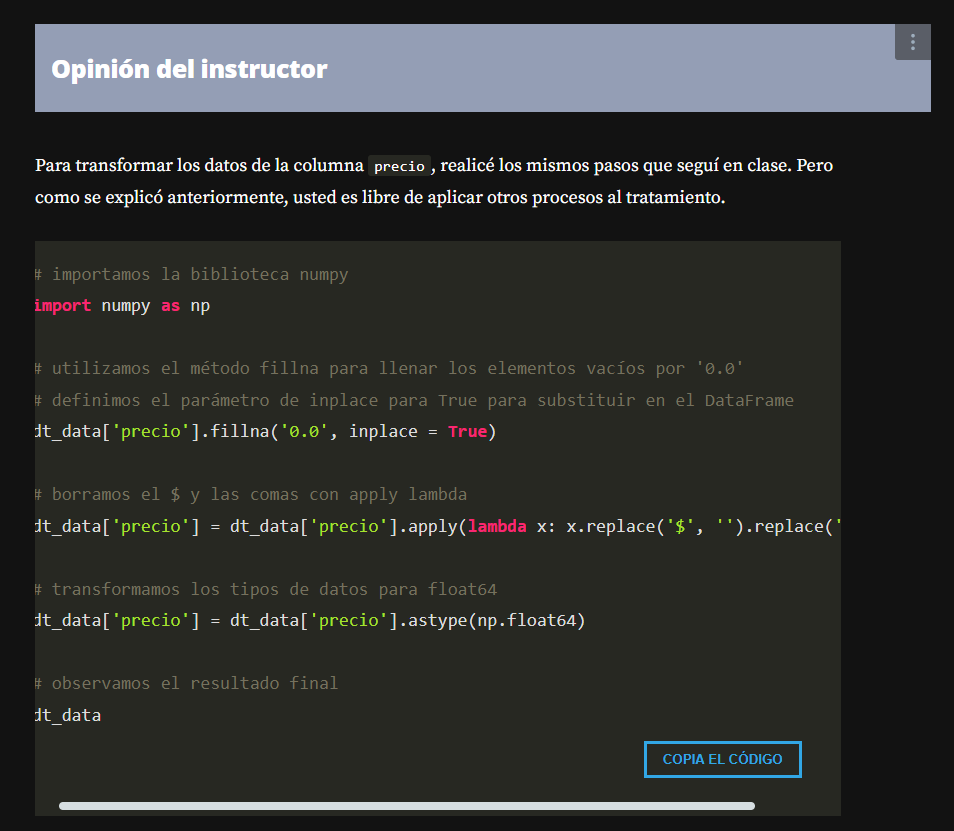
})

# Mostrando el DataFrame

print(datos)

****

****

****

# importamos la biblioteca numpy

**import** numpy **as** np

# utilizamos el método fillna para llenar los elementos vacíos por '0.0'

# definimos el parámetro de inplace para True para substituir en el DataFrame

dt\_data['precio'].fillna('0.0', inplace = **True**)

# borramos el $ y las comas con apply lambda

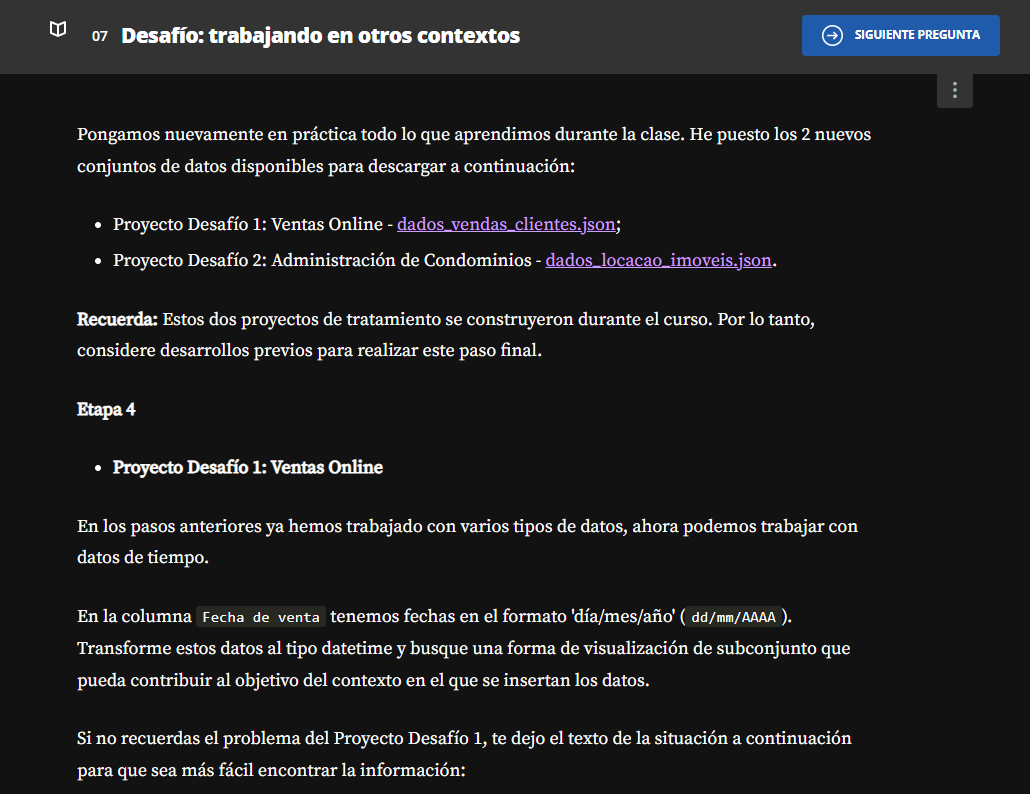
dt\_data['precio'] = dt\_data['precio'].apply(**lambda** x: x.replace('$', '').replace(',',''))

# transformamos los tipos de datos para float64

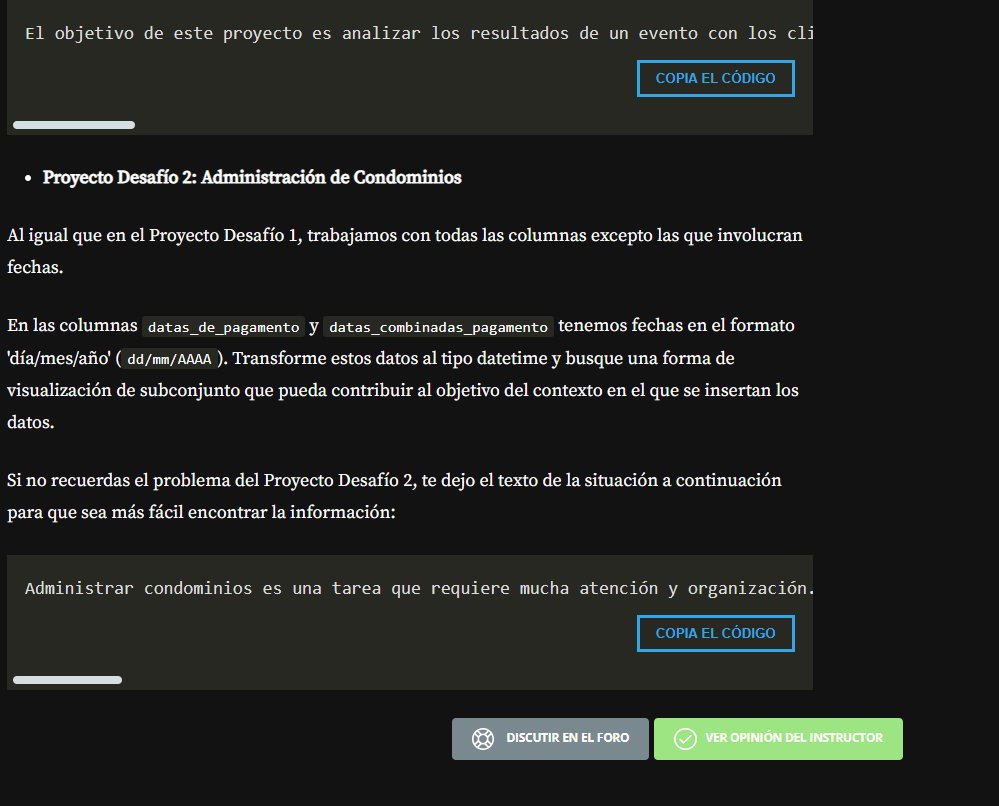
dt\_data['precio'] = dt\_data['precio'].astype(np.float64)

# observamos el resultado final

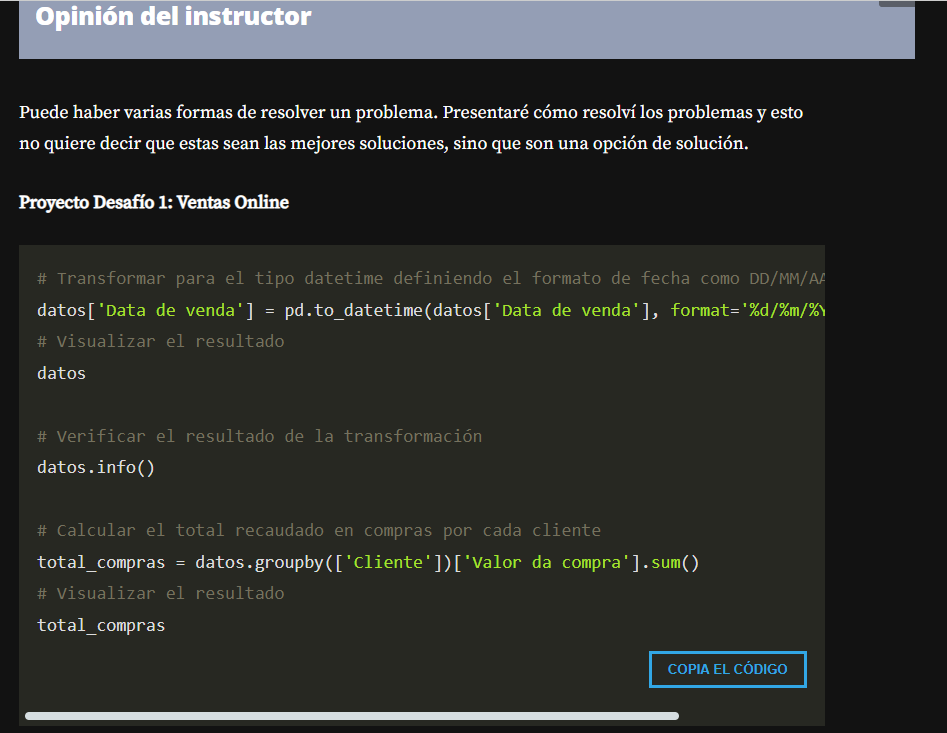
dt\_data

****

El objetivo de este proyecto es analizar los resultados de un evento con los clientes de una empresa de venta online. Se recopiló un conjunto de datos que contiene los clientes que gastaron más en productos dentro de los 5 días posteriores **a** la venta, que es la duración **del** evento. Este análisis identificará al cliente con la mayor compra esa semana, quien recibirá un premio de la tienda, y posteriormente, puede ayudar **a** la empresa **a** crear nuevas estrategias para atraer más clientes.

****

Administrar condominios es una tarea que requiere mucha atención y organización. Entre las diversas responsabilidades de gestión se encuentra el cobro **del** alquiler **a** los inquilinos. Para garantizar la buena salud financiera de la empresa, es fundamental que estos pagos se realicen de forma regular y puntual. Sin embargo, sabemos que esto no siempre sucede. Teniendo esto en cuenta, propongo un desafío de procesamiento de datos con el objetivo de analizar el retraso en el pago de la renta en el condominio ficticio de algunos residentes.

****

# Transformar para el tipo datetime definiendo el formato de fecha como DD/MM/AAAA ('%d/%m/%Y')

datos['Data de venda'] = pd.to\_datetime(datos['Data de venda'], format='%d/%m/%Y')

# Visualizar el resultado

datos

# Verificar el resultado de la transformación

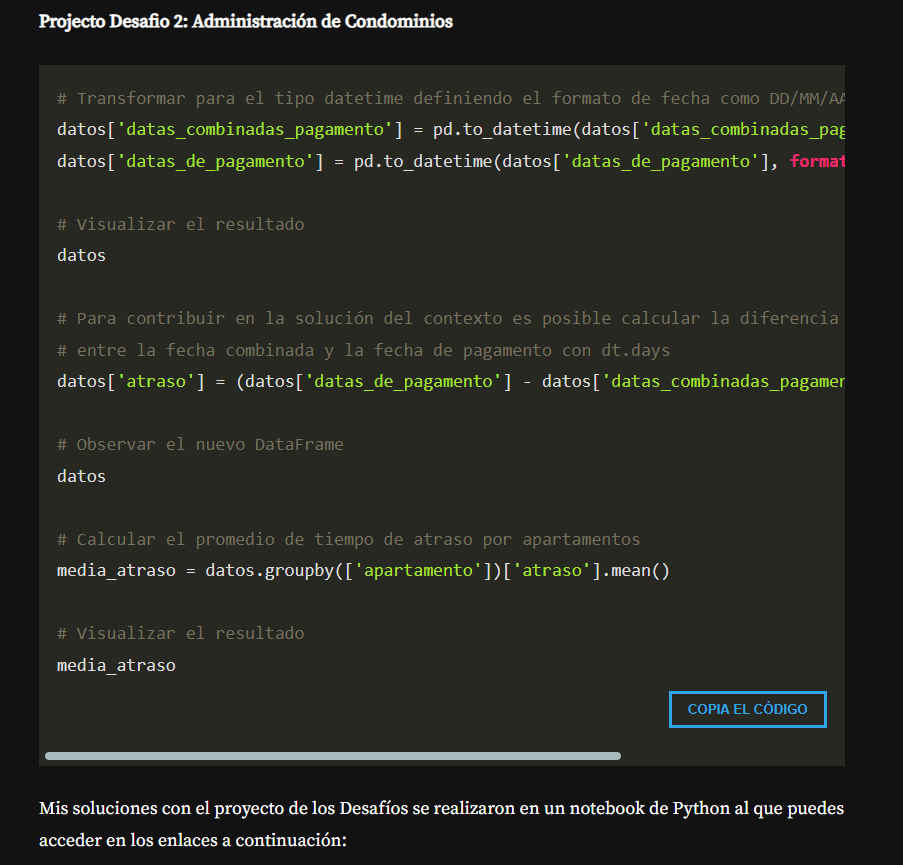
datos.info()

# Calcular el total recaudado en compras por cada cliente

total\_compras = datos.groupby(['Cliente'])['Valor da compra'].sum()

# Visualizar el resultado

total\_compras

****

# Transformar para el tipo datetime definiendo el formato de fecha como DD/MM/AAAA ('%d/%m/%Y')

datos['datas\_combinadas\_pagamento'] = pd.to\_datetime(datos['datas\_combinadas\_pagamento'], **format**='%d/%m/%Y')

datos['datas\_de\_pagamento'] = pd.to\_datetime(datos['datas\_de\_pagamento'], **format**='%d/%m/%Y')

# Visualizar el resultado

datos

# Para contribuir en la solución del contexto es posible calcular la diferencia de días

# entre la fecha combinada y la fecha de pagamento con dt.days

datos['atraso'] = (datos['datas\_de\_pagamento'] - datos['datas\_combinadas\_pagamento']).dt.days

# Observar el nuevo DataFrame

datos

# Calcular el promedio de tiempo de atraso por apartamentos

media\_atraso = datos.groupby(['apartamento'])['atraso'].mean()

# Visualizar el resultado

media\_atraso

****