

# **Manual de práctica**

Uso básico de Pandas y lectura  
de archivos.



Fundamentos de Python



## Uso básico de Pandas y lectura de archivos

### Introducción

¡Bienvenido a la práctica *Uso básico de Pandas y lectura de archivos!*

Esta actividad te permitirá conocer los conceptos fundamentales de Pandas usados para leer, manipular y analizar datos en diferentes formatos. Conocerás el proceso para trabajar con archivos .cvs y .xlsx, explorar su contenido y utilizar propiedades y métodos básicos de Pandas para procesar información de manera eficiente, a través de la resolución de un caso práctico.

¡Comencemos!

### Recursos necesarios



#### Equipo:

- Equipo de cómputo con conexión a internet.
- Windows 10, x86\_64
- 8 Gb de RAM



- Visual Studio
- Python



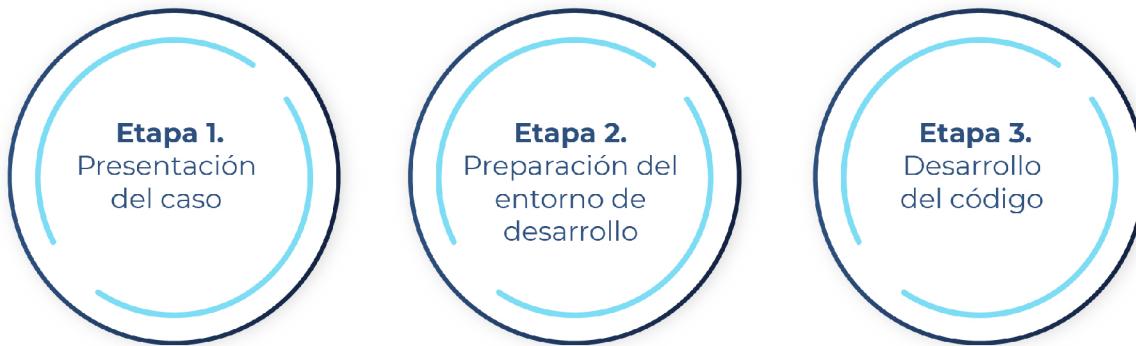
#### Archivos:

- respuestas\_correctas
- respuestas\_estudiantes



## Etapas

Antes de iniciar, es necesario que identifiques las etapas que integran esta práctica.



## Desarrollo de la práctica

### Etapa 1. Presentación del caso

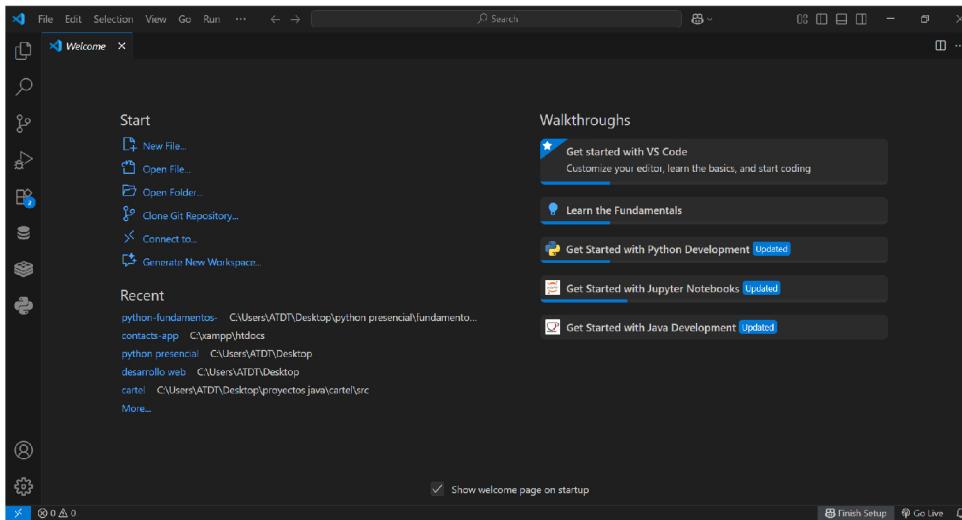
#### Calificación de exámenes

Eres profesor de un curso y necesitas **automatizar** la corrección de exámenes. Las respuestas de los estudiantes se enviaron en un archivo .csv, mientras que las respuestas correctas están en un Excel (.xlsx). Tu trabajo será desarrollar un código en Python que importe los archivos, y comparar las respuestas correctas con las de los alumnos de una serie de preguntas para agilizar la corrección de exámenes.



## Etapa 2. Preparación del entorno de desarrollo.

1. **Inicia** los programas Visual Studio y Python.



2. **Abre** la terminal o consola en Visual Studio (Control + ñ) y **escribe** los comandos `pip install pandas` y `pip install openpyxl` para instalar Pandas.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Escritorio\python\Nueva carpeta> pip install pandas
```

Puedes escribir el código `pip install pandas openpyxl` como forma simplificada de los dos anteriores:

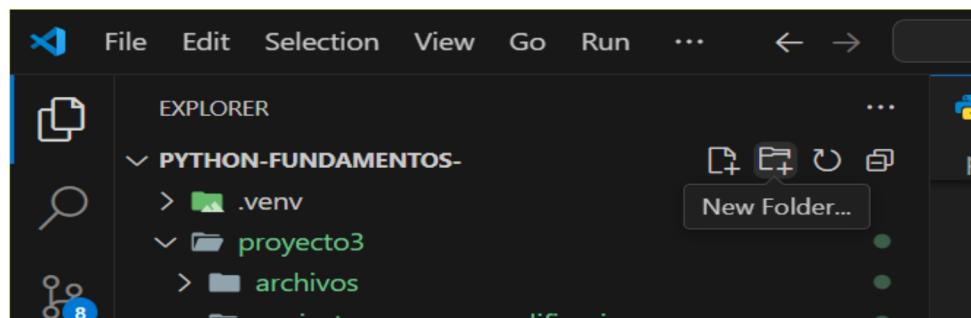
```
$ pip install pandas openpyxl
```



3. **Ejecuta** el comando presionando *Enter* y espera a que termine la instalación.

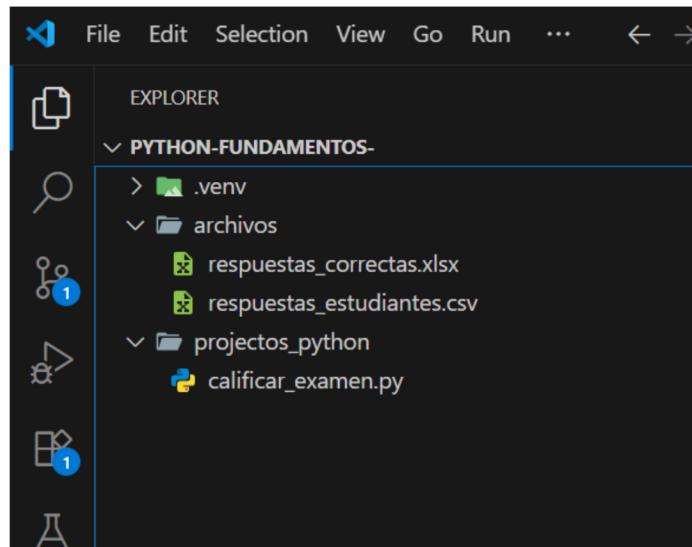
```
$ pip install pandas
Collecting pandas
  Downloading pandas-2.3.1-cp313-cp313-win_amd64.whl.metadata (19 kB)
Collecting numpy>=1.26.0 (from pandas)
  Downloading numpy-2.3.2-cp313-cp313-win_amd64.whl.metadata (60 kB)
Collecting python-dateutil>=2.8.2 (from pandas)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: tzdata>=2022.7 in c:\users\romeros\appdata\local\p
Requirement already satisfied: six>=1.5 in c:\users\romeros\appdata\local\program
Downloading pandas-2.3.1-cp313-cp313-win_amd64.whl (11.0 MB)
    11.0/11.0 MB 7.9 MB/s eta 0:00:00
Downloading numpy-2.3.2-cp313-cp313-win_amd64.whl (12.8 MB)
    12.8/12.8 MB 9.4 MB/s eta 0:00:00
Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)
Installing collected packages: pytz, python-dateutil, numpy, pandas
```

4. **Crea** una carpeta desde Visual Studio, dando clic en “*New folder*”, con el nombre “*Proyectos\_Python*”, posteriormente, **guarda** el programa con el que estás trabajando en la carpeta y asígnale el nombre *calificar\_examen.py*.





5. Descarga los 2 archivos disponibles en la plataforma (respuestas\_correctas.xlsx y respuestas\_estudiantes.csv) y guárdalos en la carpeta *Proyectos\_Python*.



### Etapa 3. Desarrollo del código

1. En el programa *calificar\_examen.py* importa la librería Pandas usando la keyword **import**, seguido del nombre de la librería **pandas**, también se usará la keyword **as**, seguido de **pd** para abbreviar el nombre de la librería. Da clic en *Enter* para ejecutar el código.

```
import pandas as pd
```



2. **Importa** el archivo *respuestas\_estudiantes.csv*, que contiene las respuestas de cada alumno, con columnas para el nombre y cada pregunta, utilizando el comando:

```
df_estudiantes = pd.read_csv("./archivos/respuestas_estudiantes.csv")
```

3. **Importa** el archivo *respuestas\_correctas.xlsx*, donde se almacenan las respuestas correctas para cada pregunta en un formato de dos columnas (pregunta y respuesta), utilizando el comando:

```
df_correctas = pd.read_excel("./archivos/respuestas_correctas.xlsx")
```

Ambos archivos se cargan en *DataFrames* de Pandas (*df\_estudiantes* y *df\_correctas*), estructuras tubulares que permiten manipulación eficiente de los datos.

```
File Edit Selection View Go Run ... ← → python-fundame
calificar_examen.py ●
projectos_python > calificar_examen.py > ...
1 import pandas as pd
2
3 # 1. Cargar los archivos
4 df_estudiantes = pd.read_csv("./archivos/respuestas_estudiantes.csv")
5 df_correctas = pd.read_excel("./archivos/respuestas_correctas.xlsx")
6
7
```



4. **Obtén** la lista de las preguntas a evaluar ['P1', 'P2', ...]) directamente de la columna 'Pregunta' del DataFrame de **df\_correctas** usando el método **.values** para convertirla en un array. Esto asegura que el sistema evalúe exactamente las preguntas definidas en la clave. Para ello, escribe el siguiente comando y da clic en *Enter*:

```
preguntas = df_correctas['Pregunta'].values
```

```
File Edit Selection View Go Run ... ← → python-fund
calificar_examen.py
projectos_python > calificar_examen.py > ...
1 import pandas as pd
2
3 # 1. Cargar los archivos
4 df_estudiantes = pd.read_csv("./archivos/respuestas_estudiantes.csv")
5 df_correctas = pd.read_excel("./archivos/respuestas_correctas.xlsx")
6
7
8 # 2. Obtener las preguntas usando métodos
9 preguntas = df_correctas['Pregunta'].values # Usando .values
10
```

5. **Crea** un diccionario de claves que almacene cada pregunta con su respuesta correcta. Para iniciar, escribe el siguiente código y da *Enter*.

```
clave_respuestas = {}
```

Con esta línea de código se crea un diccionario vacío que almacenará pares pregunta - respuesta.



6. Despues, **crea** un bucle por todas las filas del *DataFrame*, escribiendo el comando:

```
(df_correctas.shape[0]):
```

Esta línea de código devuelve el número total de preguntas, desde 0 hasta el total de preguntas existentes (10).

7. **Extrae** los datos de los dos archivos escribiendo los siguientes comandos:

```
pregunta = dfcorrectas['Pregunta'].iloc[i]  
respuestas = df=correctas['Respuestas'].iloc[i]
```

Con el primer comando, se obtendrá el nombre de la pregunta (ej. "P1") en la posición i. Mientras que con la segunda línea de código se obtiene la respuesta correspondiente a la pregunta (ej. "A").

8. Para terminar con la creación del diccionario, **escribe** el siguiente comando para almacenar cada pregunta - respuesta.

```
clave_respuestas[pregunta] = respuesta
```



```
13
14
15 # 3. Crear diccionario de respuestas correctas
16 clave_respuestas = {} # Paso 1: Creamos un diccionario vacío
17 for i in range(df_correctas.shape[0]): # Paso 2: Recorremos cada fila
18     # Paso 3: Extraemos pregunta y respuesta
19     pregunta = df_correctas['Pregunta'].iloc[i]
20     respuesta = df_correctas['Respuesta'].iloc[i]
21
22     # Paso 4: Almacenamos en el diccionario
23     clave_respuestas[pregunta] = respuesta
```

9. Para iniciar con el cálculo de puntuaciones individuales, **inicializa** una columna 'Puntuación' en cero para cada estudiante usando el siguiente código:

```
df_estudiantes['Puntuación'] = 0
```

10. Para cada pregunta:

- **Recupera** la respuesta correcta del diccionario usando las líneas de código:

```
df_estudiantes['Puntuación'] = 0
for p in preguntas:
    respuesta_correcta = clave_respuestas[p]
```

- **Compara** con la respuesta del estudiante usando operaciones vectorizadas con el código:

```
(df_estudiantes[p] == respuesta_correcta)
```



- Las comparaciones correctas (**True**) se convierten a **1** usando `astype(int)`:

```
df_estudiantes[p] == respuesta_correcta).astype(int)
```

- Se suman a las respuestas correctas usando **add**:

```
df_estudiantes['Puntuación'].add( (df_estudiantes[p] == respuesta_correcta).astype(int))
```

- El resultado es una puntuación total por estudiante en la columna '**Puntuación**'.

The screenshot shows a Jupyter Notebook interface with several icons on the left: a file, a test tube, a database, and a 3D bar chart. The code cell contains the following Python code:

```
35
36
37
38 # 4. Calcular puntuación para cada estudiante
39 df_estudiantes['Puntuación'] = 0 # Inicializa la columna de puntuación
40 for p in preguntas: # Recorre cada pregunta
41     respuesta_correcta = clave_respuestas[p] # Obtiene la respuesta correcta
42     # Compara respuestas y suma 1 punto por cada acierto:
43     df_estudiantes['Puntuación'] = df_estudiantes['Puntuación'].add(
44         (df_estudiantes[p] == respuesta_correcta).astype(int))
45
```

- 11.** Para generar el reporte detallado, primero **crea** una copia del *DataFrame* original (**df\_detalle**) para marcar las respuestas incorrectas, utilizando el código:

```
df_detalle = df_estudiantes.copy()
```

- 12.** Utiliza **.where()** para preservar las respuestas correctas. Las incorrectas se marcan añadiendo el símbolo 'X'



```
df_detalle[p] = df_detalle[p].where(  
    df_detalle[p] == clave_respuestas[p],  
    df_detalle[p]+ 'X')
```

**13. Ordena** el *DataFrame* por puntuación descendente para facilitar el análisis, haciendo uso del siguiente comando:

```
df_detalle = df_detalle.sort_values('Puntuación', ascending=False)
```

```
56 # 5. Mostrar detalle completo de respuestas  
57 df_detalle = df_estudiantes.copy() # Copia el DataFrame original  
58  
59 for p in preguntas:  
60     # Marca errores añadiendo X donde no coinciden:  
61     df_detalle[p] = df_detalle[p].where(  
62         df_detalle[p] == clave_respuestas[p],  
63         df_detalle[p] + 'X'  
64     )  
65 # Ordena por puntuación (mayor a menor):  
66 df_detalle = df_detalle.sort_values('Puntuación', ascending=False)  
67 print("Leyenda: RespuestaX = Incorrecta")  
68 print(df_detalle.to_string(index=False)) # Muestra sin índices  
69
```

**14. Muestra** una tabla ordenada con solo los nombres y puntuaciones finales, usando:

```
# 6. Mostrar resultados resumidos  
print("\n==== RESULTADOS DE LOS ESTUDIANTES ===")  
print(df_estudiantes[['Nombre', 'Puntuación']].sort_values('Puntuación', ascending=False).to_string(index=False))
```

Esto proporciona una visión clara del desempeño de cada estudiante.



15. Finalmente, para la **exportación** de los resultados, **guarda** el DataFrame con las puntuaciones en el archivo *resultados\_examen.csv*, excluyendo los índices automáticos para un formato más limpio, utilizando el siguiente comando:

```
98
99
100
101 # 7. Guardar resultados
102 df_estudiantes.to_csv("resultados_examen.csv", index=False)
103 print("\nResultados guardados en 'resultados_examen.csv'")
```

Tip: Implementa un sistema de corrección paso a paso.

**Problema común:** si se escribe todo el código junto, se vuelve confuso y difícil de depurar.

**Solución:** divide el proceso en etapas independientes, usando las siguientes líneas de código:



Para saber más

- **Carga y validación de datos:**

```
def cargar_datos(ruta_estudiantes, ruta_clave):
    -Carga archivos y verifica formatos
    -Retorna DataFrame o errores
```

- **Procesamiento central:**

```
def calcular_puntuaciones(df_estudiantes, df_clave):
    -Compara respuestas vs clave
    -Retorna DataFrame con puntuaciones
```

- **Generación de reportes:**

```
def generar_reportes(df_resultados):
    -Crea tablas de resultados y estadísticas
```

**Ventaja:** cada función resuelve un solo propósito.



Reflexión

En el caso de la práctica, cada pregunta vale 1 punto, pero no considera que algunas sean más difíciles que otras.

- ¿Cómo definir objetivamente la dificultad de cada pregunta?
- ¿Dónde debería terminar la automatización y comenzar la intervención humana?



Ejercicio de reforzamiento

Si deseas seguir practicando tus conocimientos:

Modifica el archivo CSV con tus propias respuestas para simular diferentes resultados y ver cómo afectan tu puntuación final.

**¡Listo!**

**Has concluido exitosamente la práctica sobre el uso básico de Pandas y lectura de archivos**