```
import pandas as pd
```

Introducción

En este laboratorio queremos que aprendas e investigues algunos conceptos en el contexto de Pandas: **concatenar**, **unir** y **fusionar**. Queremos revisar estos conceptos porque hará que el trabajo posterior en la transformación de los conjuntos de datos sea mucho más eficiente.

Tutorial sobre Concat, Merge y Join

Concatenando

Concatenar dos dataframes combina dos dataframes de modo que añadimos las filas de un dataframe al final del otro. Nuestros nombres de columnas tienen que ser idénticos para que esta función funcione correctamente.

A continuación, se muestra un ejemplo de la función `concat()` en pandas

https://pandas.pydata.org/docs/reference/api/pandas.concat.html

```
df1 = pd.DataFrame({'A': ['a'+str(x) for x in range(3)],
                    'B': ['b'+str(x) for x in range(3)],
                    'C': ['c'+str(x) for x in range(3)]},
                     index=[0, 1, 2])

df2 = pd.DataFrame({'A': ['a'+str(x) for x in range(3, 6)],
                    'B': ['b'+str(x) for x in range(3, 6)],
                    'C': ['c'+str(x) for x in range(3, 6)]},
                     index=[3, 4, 5])

df3 = pd.DataFrame({'D': ['d'+str(x) for x in range(3)],
                    'E': ['e'+str(x) for x in range(3)],
                    'F': ['f'+str(x) for x in range(3)]},
                     index=[0, 1, 2])

df4 = pd.DataFrame({'D': ['d'+str(x) for x in range(3, 6)],
                    'E': ['e'+str(x) for x in range(3, 6)],
                    'F': ['f'+str(x) for x in range(3, 6)]},
                     index=[3, 4, 5])

# print(df1, '\n---\n', df2, '\n---\n', df3, '\n---\n',df4)
```

Vamos a intentar concatenar `df1` y `df2`, así como `df3` y `df4`.

```
# your code here
df12=pd.concat([df1,df2])
df12
```

```
    A   B   C
0  a0  b0  c0
1  a1  b1  c1
2  a2  b2  c2
3  a3  b3  c3
4  a4  b4  c4
5  a5  b5  c5
```

```
df34=pd.concat([df3,df4])
df34
```

```
    D   E   F
0  d0  e0  f0
1  d1  e1  f1
2  d2  e2  f2
3  d3  e3  f3
4  d4  e4  f4
5  d5  e5  f5
```

Del resultado anterior, ves que el segundo dataframe se añade al final del primero.

Ahora intentemos concatenar `df1`, `df2`, `df3` y `df4` todos juntos.

Nota que se suministra el parámetro `sort=False` para silenciar un mensaje de advertencia sobre un cambio futuro en Pandas. No hace ninguna diferencia en el resultado.

```
# your code here
df1234=pd.concat([df12,df34])
df1234
```

```
     A    B    C    D    E    F
0   a0   b0   c0  NaN  NaN  NaN
1   a1   b1   c1  NaN  NaN  NaN
2   a2   b2   c2  NaN  NaN  NaN
3   a3   b3   c3  NaN  NaN  NaN
4   a4   b4   c4  NaN  NaN  NaN
5   a5   b5   c5  NaN  NaN  NaN
0  NaN  NaN  NaN   d0   e0   f0
1  NaN  NaN  NaN   d1   e1   f1
2  NaN  NaN  NaN   d2   e2   f2
3  NaN  NaN  NaN   d3   e3   f3
4  NaN  NaN  NaN   d4   e4   f4
5  NaN  NaN  NaN   d5   e5   f5
```

¿Qué encontramos?

- El método `concat` de Pandas respeta los índices de todos los ejes.

    - Debido a que `df3` y `df4` tienen índices de columnas diferentes a `df1` y `df2`, `concat` los colocó en columnas diferentes.
    - `df3` y `df4` también retienen sus índices de fila originales de 0-5 en lugar de continuar desde el último índice de `df2`.
- `concat` crea `NaN` en los lugares donde faltan valores.

Intenta también suministrar `ignore_index=True` a `concat`. ¿Cómo es diferente el resultado?

```
# tu código aquí
df1234=pd.concat([df12,df34],ignore_index=True)
df1234
```

	A	B	C	D	E	F
0	a0	b0	c0	NaN	NaN	NaN
1	a1	b1	c1	NaN	NaN	NaN
2	a2	b2	c2	NaN	NaN	NaN
3	a3	b3	c3	NaN	NaN	NaN
4	a4	b4	c4	NaN	NaN	NaN
5	a5	b5	c5	NaN	NaN	NaN
6	NaN	NaN	NaN	d0	e0	f0
7	NaN	NaN	NaN	d1	e1	f1
8	NaN	NaN	NaN	d2	e2	f2
9	NaN	NaN	NaN	d3	e3	f3
10	NaN	NaN	NaN	d4	e4	f4
11	NaN	NaN	NaN	d5	e5	f5

Fusionando y Uniendo

Pandas tiene dos funciones para unir conjuntos de datos: `merge()` y `join()`. Realizan la misma tarea pero tienen diferentes opciones y sintaxis.

A continuación, se muestra un ejemplo de `merge` y `join`. SUGERENCIA (usa la columna que se repite en ambos dataframes)

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html

```
left = pd.DataFrame({'idx': ['i'+str(x) for x in range(3)],
                     'A': ['a'+str(x) for x in range(3)],
                     'B': ['b'+str(x) for x in range(3)]})


right = pd.DataFrame({'idx': ['i'+str(x) for x in range(1,4)],
                     'C': ['c'+str(x) for x in range(1,4)],
                     'D': ['d'+str(x) for x in range(1,4)]})

left
```

	idx	A	B
0	i0	a0	b0

```
1  i1  a1  b1
2  i2  a2  b2

right

  idx   C   D
0  i1  c1  d1
1  i2  c2  d2
2  i3  c3  d3
```

`join` es idéntico a `merge`. Pero al usar join, necesitamos establecer explícitamente la columna de índice de los dataframes para unir usando `set_index`:

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.join.html

```
# tu código aquí
inner_merge = pd.merge(left, right, on='idx',
how='inner').set_index('idx')
inner_join = left.set_index('idx').join(right.set_index('idx'),
how='inner')

print("Merge:")
print(inner_merge)

print("\nJoin:")
print(inner_join)

print("\n¿Son iguales los DataFrames?")
print(inner_merge.equals(inner_join))
```

```
Merge:
      A   B   C   D
idx
i1   a1  b1  c1  d1
i2   a2  b2  c2  d2

Join:
      A   B   C   D
idx
i1   a1  b1  c1  d1
i2   a2  b2  c2  d2

¿Son iguales los DataFrames?
True
```

```
left_merge = pd.merge(left, right, on='idx',
how='left').set_index('idx')
left_join = left.set_index('idx').join(right.set_index('idx'),
how='left')

print("Merge:")
```

```
print(left_merge)

print("\nJoin:")
print(left_join)

print("\n¿Son iguales los DataFrames?")
print(left_merge.equals(left_join))
```

```
Merge:
      A   B    C    D
idx
i0   a0  b0  NaN  NaN
i1   a1  b1   c1   d1
i2   a2  b2   c2   d2

Join:
      A   B    C    D
idx
i0   a0  b0  NaN  NaN
i1   a1  b1   c1   d1
i2   a2  b2   c2   d2

¿Son iguales los DataFrames?
True
```

```
right_merge = pd.merge(left, right, on='idx',
how='right').set_index('idx')
right_join = left.set_index('idx').join(right.set_index('idx'),
how='right')

print("Merge:")
print(right_merge)

print("\nJoin:")
print(right_join)

print("\n¿Son iguales los DataFrames?")
print(right_merge.equals(right_join))
```

```
Merge:
       A    B   C   D
idx
i1    a1   b1  c1  d1
i2    a2   b2  c2  d2
i3   NaN  NaN  c3  d3

Join:
       A    B   C   D
idx
i1    a1   b1  c1  d1
i2    a2   b2  c2  d2
```

```
i3   NaN  NaN  c3  d3

¿Son iguales los DataFrames?
True
```

```
outer_merge = pd.merge(left, right, on='idx',
how='outer').set_index('idx')
outer_join = left.set_index('idx').join(right.set_index('idx'),
how='outer')

print("Merge:")
print(outer_merge)

print("\nJoin:")
print(outer_join)

print("\n¿Son iguales los DataFrames?")
print(outer_merge.equals(outer_join))
```

```
Merge:
       A    B    C    D
idx
i0    a0   b0  NaN  NaN
i1    a1   b1   c1   d1
i2    a2   b2   c2   d2
i3   NaN  NaN   c3   d3

Join:
       A    B    C    D
idx
i0    a0   b0  NaN  NaN
i1    a1   b1   c1   d1
i2    a2   b2   c2   d2
i3   NaN  NaN   c3   d3

¿Son iguales los DataFrames?
True
```

Y como ves, `join` descarta la fila de `right` con el índice no coincidente `i3`. Retiene la fila de `left` con el índice no coincidente `i0` pero usa `NaN` para los datos faltantes después de unirse.

Hay otras opciones que podemos explorar con las funciones `merge()` y `join()`.

Específicamente, podemos especificar `how`. Este argumento en la función nos indica si estamos realizando una unión interna, izquierda, derecha o externa.

También podemos especificar una columna diferente para unir en la función `merge()` usando los argumentos `left_on` y `right_on`. Consulta las siguientes documentaciones si quieres explorar más:

pandas.DataFrame.merge

pandas.DataFrame.join

Pregunta Bonus

Ahora, si miras atrás en `merge` y `join`, te das cuenta de que para realizar estas funciones en un conjunto de dataframes, estos dataframes deben compartir una columna común como índice. Solo las filas que tienen los mismos valores de índice se unirán. Esto es similar a la función `join` en MySQL, ¿no es así?

La pregunta de bonificación para ti es averiguar cómo unir y concatenar `df1`, `df2`, `df3` y `df4` que creamos al principio de este desafío. Tu producto final debería verse así:

df1-2-3-4.png

```
# tu código aquí
concat_df12_df34 = pd.concat([df12, df34], axis=1)
concat_df12_df34
```

	A	B	C	D	E	F
0	a0	b0	c0	d0	e0	f0
1	a1	b1	c1	d1	e1	f1
2	a2	b2	c2	d2	e2	f2
3	a3	b3	c3	d3	e3	f3
4	a4	b4	c4	d4	e4	f4
5	a5	b5	c5	d5	e5	f5

```
join_df12_df34 = df12.join(df34)
join_df12_df34
```

	A	B	C	D	E	F
0	a0	b0	c0	d0	e0	f0
1	a1	b1	c1	d1	e1	f1
2	a2	b2	c2	d2	e2	f2
3	a3	b3	c3	d3	e3	f3
4	a4	b4	c4	d4	e4	f4
5	a5	b5	c5	d5	e5	f5

```
merge_df12_df34 = pd.merge(df12,df34,left_index=True,right_index=True)
merge_df12_df34
```

	A	B	C	D	E	F
0	a0	b0	c0	d0	e0	f0
1	a1	b1	c1	d1	e1	f1
2	a2	b2	c2	d2	e2	f2
3	a3	b3	c3	d3	e3	f3
4	a4	b4	c4	d4	e4	f4
5	a5	b5	c5	d5	e5	f5

```
outer_merge = pd.merge(df12, df34, left_index=True, right_index=True,
how='outer')
outer_join = df12.join(df34, how='outer')
```

```python
print("Merge:")
print(outer_merge)

print("\nJoin:")
print(outer_join)

print("\n¿Son iguales los DataFrames?")
print(outer_merge.equals(outer_join))
```

```
Merge:
    A   B   C   D   E   F
0  a0  b0  c0  d0  e0  f0
1  a1  b1  c1  d1  e1  f1
2  a2  b2  c2  d2  e2  f2
3  a3  b3  c3  d3  e3  f3
4  a4  b4  c4  d4  e4  f4
5  a5  b5  c5  d5  e5  f5

Join:
    A   B   C   D   E   F
0  a0  b0  c0  d0  e0  f0
1  a1  b1  c1  d1  e1  f1
2  a2  b2  c2  d2  e2  f2
3  a3  b3  c3  d3  e3  f3
4  a4  b4  c4  d4  e4  f4
5  a5  b5  c5  d5  e5  f5

¿Son iguales los DataFrames?
True
```