06_Index

November 28, 2023

0.1 Index

0.2 Introducción

Hemos visto que tanto los objetos `Series` como `DataFrame` contienen un *índice* explícito que permite referenciar y modificar los datos. Este objeto `Index` es una estructura interesante en sí misma, y **puede ser considerada como un *array inmutable* o como un *conjunto ordenado* (técnicamente un multi-conjunto, ya que los objetos `Index` pueden contener valores repetidos)**. NOTA: un index es como una lista pero inmutable una vez creado, quizas se parece mas a una tupla pero con propiedades de un set

Como ejemplo sencillo, construyamos un `Index` a partir de una lista de enteros:

[1]:
```
import pandas as pd

ind = pd.Index([3,2,5,11,7])# un index es como un arry inmutable
ind
```

[1]:
```
Index([3, 2, 5, 11, 7], dtype='int64')
```

0.2.1 Index como array inmutable

El `Index` en muchos aspectos funciona como un array. Por ejemplo, podemos utilizar la notación de indexación estándar de Python para recuperar valores o trozos:

[2]:
```
ind[2]
```

[2]:
```
5
```

[3]:
```
#slicing
ind[2:]
```

[3]:
```
Index([5, 11, 7], dtype='int64')
```

[4]:
```
ind[::2]# de princio a fiun pero de dos en dos
```

[4]:
```
Index([3, 5, 7], dtype='int64')
```

Los objetos `Index` también tienen muchos de los atributos conocidos de las matrices de NumPy: NOTA: nosotros en principio vamos a trabajar con indices unidimensionales pero existen tb multi-dimensionales, pero no vermos mucho en el curso

[5]:
```
print(ind.ndim, ind.shape, ind.dtype,ind.size)
```

```
1 (5,) int64 5
```

Y tienen también algunos de sus métodos, pero no todos:

[6]:
```
print(ind.max())# valor max
print(ind.min())# valor min
print(ind.argmax())# el argumento max es decir la posicion dentro del indice␣
 ↪que tiene la poscion mayor
print(ind.mean())# esto dara error
```

```
11
2
3
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪1\06_Index.ipynb Celda 12 line 4
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/06_Index.ipynb#X13sZmlsZQ%3D%3D?
 ↪line=1'>2</a> print(ind.min())# valor min
      <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/06_Index.ipynb#X13sZmlsZQ%3D%3D?
 ↪line=2'>3</a> print(ind.argmax())# el argumento max es decir la posicion␣
 ↪dentro del indice que tiene la poscion mayor
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/06_Index.ipynb#X13sZmlsZQ%3D%3D?
 ↪line=3'>4</a> print(ind.mean())# esto dara error

AttributeError: 'Index' object has no attribute 'mean'
```

Una diferencia IMPORTANTE entre los objetos `Index` y las matrices de NumPy es que los índices son inmutables, es decir, no pueden ser modificados por los medios normales:

[7]:
```
ind[1]=1# dara eror porque el indice a pelo no perimite el cambio debido a su␣
↪inmutabilidad
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
↪1\06_Index.ipynb Celda 14 line 1
----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/06_Index.ipynb#X15sZmlsZQ%3D%3D?
↪line=0'>1</a> ind[1]=1

File c:
↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexes\
↪py:5347, in Index.__setitem__(self, key, value)
   5345 @final
   5346 def __setitem__(self, key, value) -> None:
-> 5347     raise TypeError("Index does not support mutable operations")

TypeError: Index does not support mutable operations
```

Esta inmutabilidad hace que sea más seguro compartir índices entre múltiples DataFrame y arrays, sin el potencial de efectos secundarios de la modificación inadvertida del índice.

0.2.2 Operaciones tipo Set

Para terminar, el objeto Index permite calcular intersecciones de índices o uniones entre índices tanto con métodos como con la syntaxis propia de los sets:

[9]:
```
indA = pd.Index([1,3,5,7,9])
indB = pd.Index([2,3,5,7,11])
```

[10]:
```
indA.intersection(indB)# los elementos que tienen en comun, creando un nuevo␣
↪indice, esto nos permitira unir tablas por indices ( veremos en la unidad␣
↪siguinete)
```

[10]:
```
Index([3, 5, 7], dtype='int64')
```

[13]:
```
indA.union(indB)# un nuevo indice con los elemento comunes y no comunes sin␣
↪repetir
```

[13]:
```
Index([1, 2, 3, 5, 7, 9, 11], dtype='int64')
```

Con esto terminamos la introduccion de las extruturas principales de los Dataframe, a partir de ahora ya operararemos con los dataframes y las series de diccionarios, pero sera en la siguient epildora