08_Seleccion_en_Series_II

November 28, 2023

0.1 Selección en Series (II)

0.1.1 Indexers: loc, iloc, ix (obsoleto o deprecated)

En esta sesión vamos a ver la forma correcta de decirle a nuestra Serie que valores queremos para evitarnos problemas. Usaremos loc e iloc.

A la hora de indexar valores en una serie puede haber confusiones si para los índices del Index de la serie hemos decidido usar enteros (o sea explicitamente hemos puesto indices con valores enteros).

Por ejemplo, si tu `Serie` tiene un índice entero explícito, una operación de indexación como **`data[1]` utilizará los índices explícitos, mientras que una operación de slicing como `data[1:3]` utilizará el índice implícito de estilo Python.** Es decir, Data[1] buscara el objeto de data que este etiquetado como 1, y data[1:3] va a buscar las posiciones 1 a 3 (no inclusive)

[1]:
```
import pandas as pd

data= pd.Series(["a","b","c"], index =[1,3,5])# esto es crear el indice␣
 ↪explicito
data
```

[1]:
```
1    a
3    b
5    c
dtype: object
```

[2]:
```
# explicit index when indexing
data[1] # aqui me da el valor que le corresponde al valor 1
```

[2]:
```
'a'
```

[3]:
```
# implicit index when slicing
data[1:3]# aqui me da los valores de la posicion 1 y 2 (es decir la 0,1=
```

[3]:
```
3    b
5    c
dtype: object
```

Debido a esta potencial confusión en el caso de los índices enteros, Pandas proporciona algunos atributos *indexadores* especiales que exponen explícitamente ciertos esquemas de indexación. No se trata de métodos funcionales, sino de atributos que exponen una interfaz de slicing particular a los datos de la `Serie`.

En primer lugar, el atributo **`loc` permite la indexación y el slice que siempre hace referencia al índice explícito:**(es decir al index)

[4]:
```
data.loc[3]
```

[4]:
```
'b'
```

[ ]:
```

```

[5]:
```
data
```

[5]:
```
1    a
3    b
5    c
dtype: object
```

[ ]:
```

```

El atributo `iloc` permite la indexación y el corte que siempre hace referencia al **índice implícito** (el posicional de "toda la vida") de estilo Python:

[15]:
```
data.iloc[3]# aqui da error pq la posicion 3 no existe, si exite el valro␣
 ↪explicito con loc 3 pero no el implicito con slicing
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
e:\Cursos\BC_Data_Science\Repositorio\ONLINE_DS_THEBRIDGE_V\SPRING 4\UNIT␣
 ↪1\08_Seleccion_en_Series_II.ipynb Celda 13 line 1

----> <a href='vscode-notebook-cell:/e%3A/Cursos/BC_Data_Science/Repositorio/
 ↪ONLINE_DS_THEBRIDGE_V/SPRING%204/UNIT%201/08_Seleccion_en_Series_II.
 ↪ipynb#X14sZmlsZQ%3D%3D?line=0'>1</a> data.iloc[3]# aqui da error pq la␣
 ↪posicion 3 no existe, si exite el valro explicito con loc 3 pero no el␣
 ↪implicito con slicing


File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexing
 ↪py:1153, in _LocationIndexer.__getitem__(self, key)
```

```
   1150 axis = self.axis or 0
   1152 maybe_callable = com.apply_if_callable(key, self.obj)
-> 1153 return self._getitem_axis(maybe_callable, axis=axis)

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexing
 ↪py:1714, in _iLocIndexer._getitem_axis(self, key, axis)
   1711     raise TypeError("Cannot index by location index with a non-integer␣
 ↪key")
   1713 # validate the location
-> 1714 self._validate_integer(key, axis)
   1716 return self.obj._ixs(key, axis=axis)

File c:
 ↪\Users\victo\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\indexing
 ↪py:1647, in _iLocIndexer._validate_integer(self, key, axis)
   1645 len_axis = len(self.obj._get_axis(axis))
   1646 if key >= len_axis or key < -len_axis:
-> 1647     raise IndexError("single positional indexer is out-of-bounds")

IndexError: single positional indexer is out-of-bounds
```

[14]:
```
data.iloc[1]
```

[14]:
```
'b'
```

[ ]:
```

```

Un tercer atributo de indexación, `ix`, es un híbrido de los dos(te buscaba por ambos metodos, por el explicito 1 y sino lo encontraba pues pasaba al explicito), y para los objetos `Series` es equivalente a la indexación estándar basada en `[]`. Está ya deprecado, es decir en las versiones de Pandas más nuevas no se puede usar, pero puede que lo encuentres en codigo de versiones previas a la 0.20. El caso es que `ix` primero buscará por etiqueta en el index y si no lo encuentra buscará como si fuera un valor posicional o implicito.

Un principio rector del código Python es que "lo explícito es mejor que lo implícito". La naturaleza explícita de `loc` y `iloc` los hace muy útiles para mantener un código limpio y legible; especialmente en el caso de los índices de enteros, **recomiendo usarlos tanto para hacer el código más fácil de leer y entender, como para prevenir errores sutiles debido a la convención de indexación/corte mixto.**

En defintiva siempre que quieras acceder a un elemento en una serie (supongamos una serie llamada "ejemplo"): * Si lo haces por una etiqueta o un valor del Index -> ejemplo.loc[valor] * si lo haces por la posición que ocupa el valor -> ejemplo.iloc[posicion] * No lo hagas así para evitar problemas ejemplo[valor] o ejemplo[posicion] y tampoco uses (te dará error) ejemplo.ix[valor], ejemplo.ix[posicion]