Introducción al objeto Index

Tanto el objeto Series como el objeto DataFrame incorporan uno o varios objetos Index que permiten especificar los ejes.

Aunque en general, la manipulación de los objetos Index se hace de forma indirecta a través de otro objetos, como por ejemplo Series o DataFrame, en este cuaderno vamos a ver una pequeña introducción a su uso directo.

Inicialización del entorno.

Lo primero será importar el paquete Pandas con alias `pd`. Posteriormente visualizamos la versión usada de Pandas ya que es un dato importante para consultar la documentación. En el momento de editar este notebook la versión de pandas es: 1.4.3

Además para facilitar los ejercicios también importamos el paquete Numpy con el alias `np`.

In [4]:
```
import pandas as pd
import numpy as np
np.set_printoptions(floatmode='fixed', precision=3)
print('Pandas versión: ', pd.__version__)
```

```
Pandas versión:  1.4.3
```

Creación directa de un índice.

Normalmente la creación de un índice se hace de forma indirecta al crear otros objetos Panda como Series o DataFrame, pero también podemos crear un índice de forma directa usando su constructor Index().

Ejercicio 01: Crear un índice para los valores dados por la lista `[-2, -1, 0, 1, 2]`.

El resultado debería ser algo parecido a lo siguiente:

```
Índice:  Int64Index([-2, -1, 0, 1, 2], dtype='int64')
```

In [5]:
```
l = [-2, -1, 0, 1, 2]
idx = None
#Pon tu código aquí.
#Sugerencia: usa el construcctor del objeto Index con la lista.

#
print('Índice: ',idx)
```

```
Índice:  None
```

Ejercicio 02: Crear un índice para los valores dados por la lista `['a', 'b', 'c', 'd', 'e']`.

El resultado debería ser algo parecido a lo siguiente:

```
Índice:  Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

In [6]:
```
l = list('abcde')
#Pon tu código aquí.
#Sugerencia: usa el construcctor del objeto Index con la lista.

#
print('Índice: ',idx)
```

```
Índice:  None
```

Ejercicio 03: Crear un índice para los valores dados por el rango entero `[0..9]`.

El resultado debería ser algo parecido a lo siguiente:

```
Índice:  RangeIndex(start=0, stop=10, step=1)
```

In [7]:
```
l = range(10)
#Pon tu código aquí.
#Sugerencia: usa el constructor del objeto Index con la lista.

#
print('Índice: ',idx)
```

```
Índice:  None
```

Como se puede ver, dependiendo del parámetro de entrada, podemos crear distintos tipos de índices, entre los cuales podemos citar RangeIndex, Int64Index, UInt64Index, Float64Index, DateTimeIndex o CategoricalIndex.

Como puedes ver en la documentación los índices para los tipos int64, uint64 y float64 serán agrupados por un sólo tipo NumericIndex a partir de la versión 2.0 de Pandas.

Veamos algunos ejemplos que crean índices de tipos específicos.

Ejercicio 04: Crear un índice no ordenado para las categorías de medios de transporte siguientes: 'bicicleta', 'patinete', 'motocicleta', 'coche', 'autobús', 'tren' y 'avion'.

El resultado debería ser algo parecido a lo siguiente:

```
Índice:  CategoricalIndex(['bicicleta', 'patinete', 'motocicleta',
'coche', 'autobús',
                  'tren', 'avion'],
                 categories=['autobús', 'avion', 'bicicleta', 'coche',
'motocicleta', 'patinete', 'tren'], ordered=False, dtype='category')
```

In [8]:
```
l = ['bicicleta', 'patinete', 'motocicleta', 'coche', 'autobús', 'tren', 'avion']

#Pon tu código aquí.
#Sugerencia: usa el construcctor del objeto CategoricalIndex con la lista.

#
print('Índice: ',idx)
```

```
Índice:  None
```

Ejercicio 05: Crear un índice temporal para los primeros cinco diás del año 2021, con inicio del día a las 00:00 y en la zona temporal "Europe/Madrid". Utiliza la función pd.date_range() para generar un rango temporal.

El resultado debería ser algo parecido a lo siguiente:

```
Índice:  DatetimeIndex(['2021-01-01 00:00:00+01:00', '2021-01-02
00:00:00+01:00',
               '2021-01-03 00:00:00+01:00', '2021-01-04 00:00:00+01:00',
               '2021-01-05 00:00:00+01:00'],
              dtype='datetime64[ns, Europe/Madrid]', freq='D')
```

In [9]:
```
#Pon tu código aquí.
#Sugerencia: usa la función pd.date_range()

#
print('Índice: ',idx)
```

```
Índice:  None
```

Index como un array inmutable.

Un objeto Index lo podemos ver como un array inmutable (que se puede usar, pero no modificar) y lo podemos indexar de forma similar a como indexamos un numpy.ndarray.

Ejercicio 06: Dado un índice queremos obtener los valores situados en posiciones impares.

El resultado debería ser algo parecido a lo siguiente:

```
Valores del índice en posiciones impares:

Index(['b', 'd'], dtype='object')
```

In [10]:
```
l = list('abcde')
idx = pd.Index(l)
print('Valores del índice en posiciones impares:\n')
#Pon tu código aquí
#Sugerencia: utiliza el índice como si fuera un ndarray de una dimensión.

#
```

```
Valores del índice en posiciones impares:
```

Además es útil en ocasiones poder consultar sus propiedades.

Ejercicio 07: Dado un índice queremos obtener sus propiedades.

El resultado debería ser algo parecido a lo siguiente:

```
Tamaño del índice: 9

Forma del índice : (9,)

Núm. dimensiones : 1

Tipo del índice  : int64
```

In [11]:
```
idx = pd.RangeIndex(1, 10, 1)
print('Tamaño del índice: ', end='')
#Pon tu código aquí
#Sugerencia usa el atributo size.
```

```
#

print('\nForma del índice : ', end='')
#Pon tu código aquí
#Sugerencia usa el atributo shape.

#

print('\nNúm. dimensiones : ', end='')
#Pon tu código aquí
#Sugerencia usa el atributo ndim.

#

print('\nTipo del índice  : ', end='')
#Pon tu código aquí
#Sugerencia usa el atributo dtype.

#
```

```
Tamaño del índice:
Forma del índice :
Núm. dimensiones :
Tipo del índice  :
```

Ejercicio 08: Dado un índice queremos saber si corresponde a un rango (es monótono) o no.

El resultado debería ser algo parecido a lo siguiente:

```
   El índice
    RangeIndex(start=1, stop=10, step=1)
       es monótono creciente? True

   El índice
    Index(['Rojo', 'Verde', 'Azul'], dtype='object')
       es monótono creciente? False
```

In [12]:
```
idx1 = pd.RangeIndex(1, 10, 1)
idx2 = pd.Index(['Rojo', 'Verde', 'Azul'])

print('El índice\n', idx1, '\n\tes monótono creciente? ', end='')
#Pon tu código aquí
#Sugerencia usa el atributo is_monotonic_increasing.

#

print('\nEl índice\n', idx2, '\n\tes monótono creciente? ', end='')
#Pon tu código aquí
#Sugerencia usa el atributo is_monotonic_increasing.

#
```

```
El índice
 RangeIndex(start=1, stop=10, step=1)
       es monótono creciente?
El índice
 Index(['Rojo', 'Verde', 'Azul'], dtype='object')
       es monótono creciente?
```

Index como conjunto de valores.

Otra forma de enteder a un objeto Index es verlo como un conjunto de valores. Esta forma de ver es

conveniente cuando se estudien las operaciones de conjunto.

Ejercicio 09: Dados índices obtener el índice que resulta de su unión.

```
Índice a:  Int64Index([-2, -1, 0, 1, 2], dtype='int64')
Índice b:  Int64Index([0, 1, 2, 3, 4], dtype='int64')
Index union:  Int64Index([-2, -1, 0, 1, 2, 3, 4], dtype='int64')
```

In [13]:
```
idx_a = pd.Index([-2, -1, 0, 1, 2])
print('Índice a: ', idx_a)
idx_b = pd.Index([0, 1, 2, 3, 4])
print('Índice b: ', idx_b)
idx_union = 0
#Pon tu código aquí.
#Sugerencia: puedes usar el operador '|' o la función DataFrame.union()

#
print('Index union: ', idx_union)
```

```
Índice a:  Int64Index([-2, -1, 0, 1, 2], dtype='int64')
Índice b:  Int64Index([0, 1, 2, 3, 4], dtype='int64')
Index union:  0
```

Ejercicio 10: Dados índices obtener el índice que resulta de su intersección.

```
Índice a:  Int64Index([-2, -1, 0, 1, 2], dtype='int64')
Índice b:  Int64Index([0, 1, 2, 3, 4], dtype='int64')
Índice intersección:  Int64Index([0, 1, 2], dtype='int64')
```

In [14]:
```
idx_a = pd.Index([-2, -1, 0, 1, 2])
print('Índice a: ', idx_a)
idx_b = pd.Index([0, 1, 2, 3, 4])
print('Índice b: ', idx_b)
idx_inter = 0
#Pon tu código aquí.
#Sugerencia: puedes usar el operador '&' o la función DataFrame.intersection()

#
print('Índice intersección: ', idx_inter)
```

```
Índice a:  Int64Index([-2, -1, 0, 1, 2], dtype='int64')
Índice b:  Int64Index([0, 1, 2, 3, 4], dtype='int64')
Índice intersección:  0
```

Otras operaciones con Index.

Hay un variado conjunto de operaciones que se pueden realizar con un objeto Index para modificarlo o realizar cálculos.

Ejercicio 11: Dado índice temporal, queremos localizar la posición del índice más cercana sin pasarse de un fecha dada.

```
Index:  DatetimeIndex(['2021-01-01', '2021-02-01', '2021-03-01', '2021-
04-01',
               '2021-05-01', '2021-06-01', '2021-07-01', '2021-08-01',
               '2021-09-01', '2021-10-01', '2021-11-01', '2021-12-01'],
              dtype='datetime64[ns]', freq='MS')
La posición del índice '2021-05-01 00:00:00' es la más cercana sin
pasarse a la fecha 2021-05-26.
```

In [15]:
```
idx = pd.date_range(start='2021-01-01', freq='MS', periods=12)
print('Index: ', idx)
fecha = '2021-05-26'
pos = -1
#Pon tu código aquí.
#Sugerencia: utliza el método asof()

#
print("La posición del índice '{}' es la más cercana sin pasarse a la fecha {}.".format(
```

```
Index:  DatetimeIndex(['2021-01-01', '2021-02-01', '2021-03-01', '2021-04-01',
               '2021-05-01', '2021-06-01', '2021-07-01', '2021-08-01',
               '2021-09-01', '2021-10-01', '2021-11-01', '2021-12-01'],
              dtype='datetime64[ns]', freq='MS')
La posición del índice '-1' es la más cercana sin pasarse a la fecha 2021-05-26.
```