Python para Análisis de Datos

Módulo 05

Pandas

Strings

Existen numerosos métodos para trabajar con strings en objetos tipo Series e Index (no aplica a Dataframes).

En general, tienen el mismo nombre que los métodos de python para strings y están agrupados en el atributo `str.` Así es posible hacer operaciones sobre todos los elementos de la estructura con un único comando.

Por ejemplo, para cambiar entre mayúsculas y minúsculas existen `lower, upper, title` y `swapcase.`

Strings

```
data["Name"].str.lower()
```

```
0                                  braund, mr. ow
1        cumings, mrs. john bradley (florence bri
2                                  heikkinen, mi
3             futrelle, mrs. jacques heath (lily
4                                  allen, mr. will
                                ...
886                                  montvila, re
887                           graham, miss. marga
888               johnston, miss. catherine helen
889                                  behr, mr. ka
890                                    dooley, mr
Name: Name, Length: 891, dtype: object
```

```
data["Name"].str.upper()
```

```
0                                  BRAUND, MR. O
1        CUMINGS, MRS. JOHN BRADLEY (FLORENCE BR
2                                  HEIKKINEN, M
3             FUTRELLE, MRS. JACQUES HEATH (LILY
4                                  ALLEN, MR. WIL
                                ...
886                                  MONTVILA, R
887                           GRAHAM, MISS. MARG
888               JOHNSTON, MISS. CATHERINE HELE
889                                  BEHR, MR. K
890                                    DOOLEY, M
Name: Name, Length: 891, dtype: object
```

Strings

El método `split` permite separar los strings en partes. Por defecto, usa el espacio en blanco como separador, pero podemos definir cualquier carácter o substring para usar como separador. Para cada elemento devuelve una lista con los strings que fueron separados.

```
data["Name"].str.split()
```

```
0                            [Braund,, Mr., Owen, Harris]
1      [Cumings,, Mrs., John, Bradley, (Florence, Bri...
2                              [Heikkinen,, Miss., Laina]
3      [Futrelle,, Mrs., Jacques, Heath, (Lily, May, ...
4                            [Allen,, Mr., William, Henry]
                               ...
886                            [Montvila,, Rev., Juozas]
887                     [Graham,, Miss., Margaret, Edith]
888       [Johnston,, Miss., Catherine, Helen, "Carrie"]
889                              [Behr,, Mr., Karl, Howell]
890                                [Dooley,, Mr., Patrick]
Name: Name, Length: 891, dtype: object
```

Strings

El método `count` cuenta cuantas veces aparece un substring en cada elemento.

```
data["Name"].str.count("Mr.")
```

```
0      1
1      1
2      0
3      1
4      1
      ..
886    0
887    0
888    0
889    1
890    1
Name: Name, Length: 891, dtype: int64
```

Strings

El método `contains` devuelve `True` o `False` según cada string contenga o no cierto substring.

```
data["Name"].str.contains("Mr.")
```

```
0       True
1       True
2      False
3       True
4       True
       ...
886    False
887    False
888    False
889     True
890     True
Name: Name, Length: 891, dtype: bool
```

Strings

Los métodos `startswith` y `endswith` permiten saber si cada string empieza o termina con cierto prefijo o sufijo.

```
#cantidad de strings que empiezan con "Andersson"
data["Name"].str.startswith("Andersson").sum()
```

```
9
```

```
# Existe algún string que empieze con "Pérez"?
data["Name"].str.startswith("Pérez").any()
```

```
False
```

Strings

El método `len` calcula la cantidad de caracteres de cada string, como si estuviéramos aplicando la función `len` a cada uno.

```
data["Name"].str.len()
```

```
0      23
1      51
2      22
3      44
4      24
       ..
886    21
887    28
888    40
889    21
890    19
Name: Name, Length: 891, dtype: int64
```

Strings

También ha varios métodos para comprobar si los strings están compuestos por cierto tipo de caracteres.

Los métodos `isdigit, isnumeric` e `isdecimal` permiten chequear que sean caracteres numéricos; `isalpha` que sean letras; `isalnum` que sean letras o números; `islower, isupper` e `istitle` que cumplan con el formato.

```
s = pd.Series(["abc", "ABC", "007", "abc007"])
```

```
s.str.isalpha()
```

```
0     True
1     True
2    False
3    False
dtype: bool
```

```
s.str.islower()
```

```
0     True
1    False
2    False
3     True
dtype: bool
```

```
s.str.isnumeric()
```

```
0    False
1    False
2     True
3    False
dtype: bool
```

Strings

El atributo `str` también permite hacer indexación y slices sobre los strings usando corchetes.

En caso de indexar una posición mayor a la longitud del string se devuelve `NaN`.

```
data["Name"].str[0:3]
```

```
0      Bra
1      Cum
2      Hei
3      Fut
4      All
      ...
886    Mon
887    Gra
888    Joh
889    Beh
890    Doo
Name: Name, Length: 891
```

```
data["Name"].str[40]
```

```
0      NaN
1        g
2      NaN
3        e
4      NaN
      ...
886    NaN
887    NaN
888    NaN
889    NaN
890    NaN
Name: Name, Length: 891,
```

¡Muchas gracias!

¡Sigamos trabajando!