PANDAS
ESTRUCTURAS DE DATOS

Crear una Serie

A partir de un arreglo de Numpy

Sin indicar los índices

```
In [ ]: s = pd.Series(np.random.randn(5))
In [ ]: s
Out[ ]:
0 0.604759
1 0.053896
2 -0.756650
3 -0.212134
4 1.883489
dtype: float64

In [ ]: s.index
Out[ ]: RangeIndex(start=0, stop=5, step=1)
```

Indicando los índices

```
In [ ]: s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])
In [ ]: s
Out[ ]:
a 0.465723
b -1.220388
c -0.032099
d -1.335270
e -0.156012
dtype: float64

In [ ]: s.index
Out[ ]: Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

A partir de un diccionario

```
In [ ]: d = {'b' : 1, 'a' : 0, 'c' : 2}
In [ ]: s1 = pd.Series(d)
In [ ]: s1
Out[ ]:
a 0
b 1
c 2
dtype: int64

In [ ]: s1.index
Out[ ]: Index(['a', 'b', 'c'], dtype='object')
```

A partir de un escalar

```
In [ ]: pd.Series(5., index=['a', 'b', 'c', 'd', 'e'])
Out[ ]:
a 5.0
b 5.0
c 5.0
d 5.0
e 5.0
dtype: float64
```

Operando Series como arreglos de Numpy

```
In [ ]: s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])

In [ ]: s
Out[ ]:
a 1.512588
b -1.459946
c 0.524543
d 2.354224
e 0.785034
dtype: float64

In [ ]: s[1]
Out[ ]: -1.4599457192384637
```

```
In [ ]: s[:3]
Out[ ]:
a 1.512588
b -1.459946
c 0.524543
dtype: float64

In [ ]: s[[4, 3, 1]]
Out[ ]:
e 0.785034
d 2.354224
b -1.459946
dtype: float64

In [ ]: s[s > s.median()]
Out[ ]:
a 1.512588
d 2.354224
dtype: float64

In [ ]: np.exp(s)
Out[ ]:
a 4.538463
b 0.232249
c 1.689686
d 10.529949
e 2.192480
dtype: float64

In [ ]: s + s
Out[ ]:
a 3.025177
b -2.919891
c 1.049086
d 4.708447
e 1.570067
dtype: float64

In [ ]: s*2
Out[ ]:
a 3.025177
b -2.919891
c 1.049086
d 4.708447
e 1.570067
dtype: float64
```

```
In [38]: s.append(pd.Series(7, index=['f']))
Out[38]:
a 1.512588
b -1.459946
c 0.524543
d 2.354224
e 0.785034
f 7.000000
dtype: float64
```

Operando Series como Diccionarios

```
In [ ]: s['a']
Out[ ]: 1.5125883769478499

In [ ]: 'a' in s
Out[ ]: True

In [ ]: 'f' in s
Out[ ]: False

In [ ]: s['g']=3
In [ ]: s
Out[ ]:
a 1.512588
b -1.459946
c 0.524543
d 2.354224
e 0.785034
g 3.00000
dtype: float64
```

Salvando a un csv

```
s.to_csv('serie.csv')
```

Crear un DataFrame

A partir de un diccionario

```
In [ ]: d = {'Código': [20152300120, 20153300123, 20172400322, 20172400436],
'Nota1': [3.3, 4.1, 1.5, 2.0], 'Nota2': [2.1, 3.8, 3.5, 3.6], 'Nota3': [3.3, 4.1, 1.5, 4.1] }
In [ ]: df = pd.DataFrame(data=d)

In [ ]: df
Out[ ]:
  Código            Nota 1  Nota2  Nota3
0 20152300120        3.3    2.1    3.3
1 20153300123        4.1    3.8    4.1
2 20172400322        1.5    3.5    1.5
3 20172400436        2.0    3.6    4.1
```

Estableciendo índices

```
In [ ]: df = df.set_index('Código')
In [ ]: df
Out[ ]:
Nota1 Nota2 Nota3
Código
20152300120 3.3 2.1 3.3
20153300123 4.1 3.8 4.1
20172400322 1.5 3.5 1.5
20172400436 2.0 3.6 4.1

In [ ]: df.index
Out[ ]: Int64Index([20152300120, 20153300123, 20172400322, 20172400436],
dtype='int64', name='Código')

In [ ]: df.columns
Out[ ]: Index(['Nota1', 'Nota2', 'Nota3'], dtype='object')
```

Agregando índices

```
In [ ]: d = {'one' : [1., 2., 3., 4.],'two' : [4., 3., 2., 1.]}

In [ ]: df1= pd.DataFrame(d)
```

```
Out[ ]:
one two
0 1.0 4.0
1 2.0 3.0
2 3.0 2.0
3 4.0 1.0
```

```
In [ ]: df2= pd.DataFrame(d, index=['a', 'b', 'c', 'd'])
Out[ ]:
one two
a 1.0 4.0
b 2.0 3.0
c 3.0 2.0
d 4.0 1.0
```

```
In [ ]: df1 = df1.set_index([['a','b','c','d']])
Out[]:
   one  two
a  1.0  4.0
b  2.0  3.0
c  3.0  2.0
d  4.0  1.0
```

Creando DF de Diccionarios de Series

```
In [ ]: d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),'two' : pd.Series([1., 2., 3., 4.],
index=['a', 'b', 'c', 'd'])}
```

```
In [ ]: df = pd.DataFrame(d)
```

```
In [ ]: df
Out[ ]:
one two
a 1.0 1.0
b 2.0 2.0
c 3.0 3.0
d NaN 4.0
```

```
In [ ]: pd.DataFrame(d, index=['d', 'b', 'a'])
Out[ ]:
one two
d NaN 4.0
b 2.0 2.0
a 1.0 1.0
```

```
In [ ]: pd.DataFrame(d, index=['d', 'b', 'a'], columns=['two', 'three'])
Out[ ]:
two three
d 4.0 NaN
b 2.0 NaN
a 1.0 NaN
```

Creando DF de Listas de Diccionarios

```
In [ ]: data2 = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]

In [ ]: pd.DataFrame(data2)
Out[ ]:
a b c
0 1 2 NaN
1 5 10 20.0

In [ ]: pd.DataFrame(data2, index=['first', 'second'])
Out[ ]:
a b c
first 1 2 NaN
second 5 10 20.0

In [ ]: pd.DataFrame(data2, columns=['a', 'b'])
Out[ ]:
a b
0 1 2
1 5 10
```

Salvar a csv

```
In [ ]: df.to_csv('notas.csv')
```

A partir de un arreglo de Numpy

```
In [ ]: a = np.random.randint(low=0, high=10, size=(5, 5))
In [ ]: a
Out[ ]:
Array ([[1, 1, 6, 4, 4],
        [7, 4, 1, 7, 5],
        [8, 0, 1, 6, 7],
        [7, 2, 6, 4, 5],
        [0, 4, 7, 2, 5]])
In [ ]: df2 = pd.DataFrame(data=a)
In [ ]: df2
Out[ ]:
  0 1 2 3 4
0 1 1 6 4 4
1 7 4 1 7 5
2 8 0 1 6 7
3 7 2 6 4 5
4 0 4 7 2 5
```

Definiendo las Columnas

```
In [28]: df2 = pd.DataFrame(data=a, columns=['punt1', 'punt2', 'punt3', 'punt4', 'punt5'])
In [29]: df2
Out[29]:
        punt1   punt2   punt3   punt4   punt5
0       1       1       6       4       4
1       7       4       1       7       5
2       8       0       1       6       7
3       7       2       6       4       5
4       0       4       7       2       5
```

Definiendo los Índices

```
In [34]: df2 = pd.DataFrame(data=a, columns=['punt1', 'punt2', 'punt3', 'punt4', 'punt5'], index
= ['est1', 'est2', 'est3', 'est4', 'est5'])

In [35]: df2
Out[35]:
        punt1   punt2   punt3   punt4   punt5
est1    1       1       6       4       4
est2    7       4       1       7       5
est3    8       0       1       6       7
est4    7       2       6       4       5
est5    0       4       7       2       5
```

Renombrando Índices y Columnas

```
In [ ]: df2.rename(columns={0: "a", 1: "b", 2: "c", 3: "d", 4: "e"})
Out[ ]:
  a  b  c  d  e
```

```
0  3  1  6  2  4
1  1  2  6  3  9
2  6  2  0  1  1
3  3  8  7  0  2
4  0  8  3  1  0
```

```
In [ ]: df2.rename(index={0: "a", 1: "b", 2: "c", 3: "d", 4: "e"})
Out[ ]:
   0  1  2  3  4
a  3  1  6  2  4
b  1  2  6  3  9
c  6  2  0  1  1
d  3  8  7  0  2
e  0  8  3  1  0
```

A partir de un csv (Taller)

https://www.datos.gov.co/ -> Descubre -> Docentes de planta

```
In [ ]: docentes = pd.read_csv('Docentes_De_Planta.csv')
```

```
In [ ]: docentes.head()
In [ ]: docentes.tail()
In [ ]: docentes.columns
In [ ]: docentes.index
In [ ]: docentes.describe()
```

1. Consular los tipos de datos de cada una de las columnas

```
In [ ]: docentes.dtypes
Out[ ]:
Facultad                    object
Programa Académico          object
TC                          float64
MT                          int64
TOTAL                       int64
PREGRADO                    float64
ESPECIALIZACIÓN             float64
MAESTRÍA                    float64
DOCTORADO                   float64
TOTAL G                     int64
AUXILIAR                    float64
ASISTENTE                   float64
ASOCIADO                    float64
```

```
TITULAR                    float64
TOTAL GENERAL              int64
```

2. Configurar un índice

```
In [ ]: docentes = pd.read_csv('Docentes_De_Planta.csv', index_col = 'Facultad')
```

3. Obtener una columna completa

```
In [ ]: docentes['DOCTORADO']
```

4. Obtener los primeros 10 registros de 3 columnas

```
In [ ]: docentes[['DOCTORADO','ASISTENTE','TITULAR']][:10]
In [ ]: docentes[['DOCTORADO','ASISTENTE','TITULAR']].head(10)
```

Ojo -> .loc   iloc

```
docentes.loc[´fila´]
Docentes.loc[´fila2:fila4´]
docentes.iloc[0:3],
df1.iloc[[1, 3, 5], [1, 3]]
```

5. Obtener los últimos 15 registros de 2 columnas

```
In [ ]: docentes[['ASISTENTE','TITULAR']][-15:]
In [ ]: docentes[['ASISTENTE','TITULAR']].tail(15)
```

6. Graficar todo el DataFrame, cambiar aspecto de la gráfica (ej: tamaño, título,...)

```
In [ ]: docentes.plot(figsize=(10,15))
```

7. Graficar una sola columna

```
In []: docentes['DOCTORADO'].plot(figsize=(10,15))
```

8. Encontrar la frecuencia de los valores de una columna y graficar dicha frecuencia mediante un diagrama de barras.

```
In [ ]: docentes['DOCTORADO'].value_counts().plot(kind = 'bar')
```

Adicionar datos

```
In [ ]: docentes['HONORIS CAUSA'] = range(39)
```

```
In [ ]: docentes['HONORIS CAUSA'] = docentes['TITULAR']
```