Existen 3 tipos de datos en pandas:

```
 - Series --> Son las series de elementos en una dimensión, como pueden ser las tuplas, las listas o los diccionarios.
- DataFrames --> Son los sets de datos de 2 dimiensiones, es decir, tablas.
-Index objects --> Se usan para guardar metadata de otras estructuras de datos.
```

In [16]:
```python
import pandas as pn
import numpy as np
```

SERIES

Se definen como **pn.Series()**, por ejemplo:

In [3]:
```python
Serie1=pn.Series([1,2,3])
Serie1
```

Out[3]:
```
0    1
1    2
2    3
dtype: int64
```

Podemos crear un diccionario en el que las **keys serán los índices y los values los valores de la serie**:

In [4]:
```python
dic={'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e': 800}
serieA=pn.Series(dic)
serieA
```

Out[4]:
```
a    100
b    200
c    300
d    400
e    800
dtype: int64
```

serie.values , serie.index

In [5]:
```python
Serie1.values  #Permite ver el contenido de la serie
```

Out[5]:
```
array([1, 2, 3], dtype=int64)
```

In [6]:
```python
Serie1.index #Permite ver el índice de los elementos de la serie
```

Out[6]:
```
RangeIndex(start=0, stop=3, step=1)
```

Cómo llamar a elementos concretos de una serie

In [4]:
```python
Serie2=pn.Series([1,2,3,4,5], index=['a','b','c','d','e']) #Permite indexar los elementos del array según éste es creado
```

In [8]:
```python
Serie2.index #Permite ver los índices de la serie indicada
```

Out[8]:
```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

In [9]:
```python
Serie2['b'] #Como en Python, para acceder a un valor concreto se pone entre corchetes,
#la diferencia es que en este caso también puedes acceder por el índice
```

Out[9]:
```
2
```

In [10]:
```python
Serie2[1]
```

Out[10]:
```
2
```

In [7]:
```python
Serie2[['a','b','c']] #Para acceder a varios datos debes poner doble corchete [[]]
```

Out[7]:
```
a    1
b    2
c    3
dtype: int64
```

In [12]:
```python
Serie2['b']=4 #Puedes cambiar los valores como en Python
Serie2['b']
```

Out[12]:
```
4
```

In [13]:
```python
Serie2[Serie2<=3] #Puedes introducir Booleanos en los corchetes
```

Out[13]:
```
a    1
c    3
dtype: int64
```

In [14]:
```python
Serie3=pn.Series({'Pepe':19,'Juan':21,'Luis':30,'Paco':16}) #También funciona para diccionarios
Serie3
```

Out[14]:
```
Pepe    19
Juan    21
Luis    30
Paco    16
dtype: int64
```

In [15]:
```python
Serie3=pn.Series({'Pepe':19,'Juan':21,'Luis':30,'Paco':None}, index=['Pepe','Juan','Luis','Paco']) #También podemos poner ,index=[] en un diccionario, pero como el object ya es un index de por sí es un poco tontería
Serie3
```

Out[15]:
```
Pepe    19.0
Juan    21.0
Luis    30.0
Paco     NaN
dtype: float64
```

pn.isnull() y pn.notnull()

In [8]:
```python
pn.isnull(Serie2) #Te enseña qué elementos de la serie son nulos, si es nulo pondrá True y si no lo es pondrá False
```

Out[8]:
```
a    False
b    False
c    False
d    False
e    False
dtype: bool
```

In [17]:
```python
pn.notnull(Serie2) #Lo opuesto a pn.isnull(), te muestra los elementos de la serie que no son nulos con True, y los que son nulos con False
```

Out[17]:
```
a    True
b    True
c    True
d    True
e    True
dtype: bool
```

Un concepto importante de pandas es que a la hora de hacer operaciones aritméticas sobre series hace un alineamiento automático, es decir, busca los índices que son iguales y los fusiona, y si hay algún índice que no coincide se le asigna valor **NaN**, independientemente de si antes tenía un valor o no.

DATAFRAMES

Crear un dataframe

Se crean igual que una serie, pero es necesario poner pn.DataFrame() y la tabla se crea como un diccionario en el que la key es el **título de la columna (categoría)** y el value es **una tupla en la que cada valor es una fila**

In [16]:
```python
data={'state':['Ohio','Ohio','Ohio','Nevada','Nevada'],'year':[2000,2001,2002,2001,2002],'pop':[1.5,1.7,3.6,2.4,2.9]}
frame=pn.DataFrame(data)
frame
```

Out[16]:

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9

Añadiendo **,columns=['X','Y','Z'...] podemos cambiar el orden o título de las categorías** o añadir nuevas categorías, pero éstas al no haberle metido ningún valor tendrán valor NaN.

In [19]:
```python
frame2=pn.DataFrame(data, columns=['pop','state','year'])
frame2
```

Out[19]:

	pop	state	year
0	1.5	Ohio	2000
1	1.7	Ohio	2001
2	3.6	Ohio	2002
3	2.4	Nevada	2001
4	2.9	Nevada	2002

Añadiendo **,index['one','two','three',...]** podemos cambiar los índices (números que aparecen a la izquierda de la tabla), que por defecto son 0,1,2,3,4...

In [34]:
```python
frame3=pn.DataFrame(data, columns=['pop','state','year'], index=['one','two','three','four','five'])
frame3
```

Out[34]:

	pop	state	year
one	1.5	Ohio	2000
two	1.7	Ohio	2001
three	3.6	Ohio	2002
four	2.4	Nevada	2001
five	2.9	Nevada	2002

frame.values, **frame.columns** y **frame.index**

In [21]:
```python
frame3.values #Te muestra los valores del dataframe creado
```

Out[21]:
```
array([[1.5, 'Ohio', 2000],
       [1.7, 'Ohio', 2001],
       [3.6, 'Ohio', 2002],
       [2.4, 'Nevada', 2001],
       [2.9, 'Nevada', 2002]], dtype=object)
```

In [22]:
```python
frame3.columns #Te muestra los nombres de las columnas, es decir el título de cada columna, es decir, las categorias
```

Out[22]:
```
Index(['pop', 'state', 'year'], dtype='object')
```

In [23]:
```python
frame3.index #Te muestra los indices (por defecto son 0,1,2,3...)
```

Out[23]:
```
Index(['one', 'two', 'three', 'four', 'five'], dtype='object')
```

Para llamar a elementos de un dataframe se hace de la misma forma que para llamar a elementos de una serie, pero teniendo en cuenta que primero llamas a un diccionario y luego a una tupla indexada, por ello:

In [24]:
```python
frame3['pop'] #LLamamos a la columna pop al completo
```

Out[24]:
```
one      1.5
two      1.7
three    3.6
four     2.4
five     2.9
Name: pop, dtype: float64
```

In [35]:
```python
frame3['pop']['three'] #LLamamos específicamente al elemento de la columna pop con indice three
```

Out[35]:
```
3.6
```

Coger una fila completa en función del índice con frame.loc[indice]

In [26]:
```python
fram4=frame3.loc['three'] #Te permite meter en una variable la fila de indice 'three' con sus correspondientes categorías, que pasan a ser los indices de esta serie creada
fram4
```

Out[26]:
```
pop        23
state    Ohio
year     2002
Name: three, dtype: object
```

Añadir categorías (columnas) a un dataframe ya creado

Para ello debemos de llamar a un dataframe con un índice que no exista, de forma que se creará esta nueva categoría y se llenará con los valores que especifiquemos:

In [33]:
```python
import numpy as np
frame3['categoria_añadida']=np.nan
frame3
```

Out[33]:

	pop	state	year	categoria_añadida
one	1.5	Ohio	2000	NaN
two	1.7	Ohio	2001	NaN
three	3.6	Ohio	2002	NaN
four	2.4	Nevada	2001	NaN
five	2.9	Nevada	2002	NaN

IMPORTANTE: CAMBIAR EL VALOR DE UN ELEMENTO CONCRETO DE LA TABLA

TABLA.LOC['INDICE','CATEGORIA] Es muy importante que pongas coma y no separes los elementos en 2 corchetes, porque si lo haces no te va a dejar cambiarlo.

Por ejemplo si quieremos cambiar el año de la fila con índice five:

In [28]:
```python
frame3.loc['five','year']=2021
frame3
```

Out[28]:

	pop	state	year	categoria_añadida
one	1.5	Ohio	2000	None
two	1.7	Ohio	2001	None
three	23.0	Ohio	2002	None
four	2.4	Nevada	2001	None
five	2.9	Nevada	2021	None

Forma alternativa de crear dataframes

Podemos establecer un data frame con un **diccionario de diccionarios**, en el que el primer key es el título de la columna, y el value es un diccionario en el que a su vez distinguimos keys (que serán el **índice** de cada fila) y los values serán los valores de la columna. Por ello podemos crear:

In [29]:
```python
frame4=pn.DataFrame({'pop':{'one':1.2,'two':2.3,'three':4.5,'four':5.6,'five':6.7},'state':{'one':'Ohio','two':'Ohio','three':'Ohio','four':'Nevada','five':'Nevada'},'year':{'one':2000,'two':2001,'three':2002,'four':2001,'five':2
frame4
```

Out[29]:

	pop	state	year
one	1.2	Ohio	2000
two	2.3	Ohio	2001
three	4.5	Ohio	2002
four	5.6	Nevada	2001
five	6.7	Nevada	2002

EJERCICIO

- You are working on a project for data analysis related with the automovile sector. In order to do some tests you plan to define some test data
- Think about the structure of DataFrame for storing the relevant data of the car models of a company
- Once defined the structure, fill in the DataFrame with some test data using any of the methods explained for DataFrame creation
- Change the DataFrame index and assign a new one, for example changing the data type (e.g. int to string)

In [28]:
```python
frame_coches=pn.DataFrame({'matricula':['2345AVV','3333AAA','4444VVV','4444BBB'],'Marca':['Renault','Skoda','Peugeot','Seat'],'Gama':[1,2,2,2],'color':['Azul','Verde','Verde','Gris'],'Precio':[40,10,20,15],'Mano':[1,1,1,1]})
frame_coches

frame_coches.loc[frame_coches.Gama>1]
```

Out[28]:

	matrícula	Marca	Gama	color	Precio	Mano
1	3333AAA	Skoda	2	Verde	10	1
2	4444VVV	Peugeot	2	Verde	20	1
3	4444BBB	Seat	2	Gris	15	1

In [31]:
```python
frame_coches=pn.DataFrame(columns=['Matrícula','Marca','Gama','Color','Precio','Mano'])
frame_coches
```

Out[31]:

Matrícula	Marca	Gama	Color	Precio	Mano

In [32]:
```python
frame_coches['Matrícula']=['2345AVV','3333AAA','4444VVV','4444BBB']
frame_coches['Marca']=['Renault','Skoda','Peugeot','Seat']
frame_coches['Gama']=[1,2,2,2]
frame_coches['Color']=['Azul','Verde','Verde','Gris']
frame_coches['Precio']=[40,10,20,15]
frame_coches['Mano']=[1,1,1,1]
frame_coches
```

Out[32]:

	Matrícula	Marca	Gama	Color	Precio	Mano
0	2345AVV	Renault	1	Azul	40	1
1	3333AAA	Skoda	2	Verde	10	1
2	4444VVV	Peugeot	2	Verde	20	1
3	4444BBB	Seat	2	Gris	15	1

In [33]:
```python
frame_coches.index=['a','b','c','d']
frame_coches
```

Out[33]:

	Matrícula	Marca	Gama	Color	Precio	Mano
a	2345AVV	Renault	1	Azul	40	1
b	3333AAA	Skoda	2	Verde	10	1
c	4444VVV	Peugeot	2	Verde	20	1
d	4444BBB	Seat	2	Gris	15	1

In [34]:
```python
serieA=pn.Series([2, 4, 6, 8, 10])
serieB=pn.Series([1, 3, 5, 7, 10])

suma=serieA+serieB
resta=serieA-serieB
multiplicacion=serieA*serieB
potencia=serieA**serieB
print(suma,resta,multiplicacion,potencia)
```

```
0     3
1     7
2    11
3    15
4    20
dtype: int64 0    1
1    1
2    1
3    1
4    0
dtype: int64 0      2
1     12
2     30
3     56
4    100
dtype: int64 0              2
1             64
2           7776
3        2097152
4    10000000000
dtype: int64
```

Ejercicio 2: Crear una serie a partir de este diccionario

In [35]:
```python
dic={'a': 100, 'b': 200, 'c': 300, 'd': 400, 'e': 800}
values=[]
keys=[]
for i in dic.values():
    values.append(i)
for i in dic.keys():
    keys.append(i)
serieA=pn.Series(values,index=keys)
serieA=pn.Series(dic)
serieA
```

Out[35]:
```
a    100
b    200
c    300
d    400
e    800
dtype: int64
```

Ejercicio 3: Write a Pandas program to convert the first column of a DataFrame as a Series and then convert that Series to an Array

In [36]:
```python
data_frame=pn.DataFrame(columns=['col1','col2','col3'])
data_frame['col1']=[1,2,3,4,7,11]
data_frame['col2']=[4,5,6,9,5,0]
data_frame['col3']=[7,5,8,12,1,11]

columna1=data_frame['col1']
columna1 #Esta es la serie de Pandas

array_python=columna1.values
array_final_python=[]
for i in array_python:
    array_final_python.append(i)
array_final_python
```

Out[36]:
```
[1, 2, 3, 4, 7, 11]
```

Ejercicio 4: Write a Pandas program to get the items of a given series not present in another given series.

In [22]:
```python
serie1=pn.Series([1,2,3,4,7,11])
serie2=pn.Series([4,5,6,7,9,5,11,0])

lista=[]
long=len(serie2)
for i in serie1:
    cont=0
    for j in serie2:
        if i != j:
            cont=cont+1
    if cont == long:
        lista.append(i)
serie4=pn.Series(lista)
serie4
```

Out[22]:
```
0    1
1    2
2    3
dtype: int64
```

Ejercicio 5: Write a Pandas program to compute the minimum, 25th percentile, median, 75th, and maximum of a given series.

In [38]:
```python
import numpy as np
serie2=pn.Series([4,5,6,7,9,5,11,0])
minu=min(serie2)
print(minu)

print (max(serie2))

print (np.percentile(serie2,25))
print (np.percentile(serie2,50))
print (np.percentile(serie2,75))
```

```
0
11
4.75
5.5
7.5
```

Ejercicio 6: Write a Pandas program to display most frequent value in a given series and replace everything else as 'Other' in the series.

In [39]:
```python
serie=pn.Series([4,5,6,7,9,5,11,0,12,34,3,4,5,6,2,1,3,4,3,4,3,5,6,6,6,6,7,45,6,7,4,1,1])
max=0
for i in serie:
    cont=0
    for j in serie:
        if i ==j:
            cont=cont+1
        if cont > max:
            max=cont
            most_frecuent=i

cont2=0
for i in serie:
    if i != most_frecuent:
        serie[cont2]='Other'
    cont2+=1

print (serie)
```

```
0     Other
1     Other
2         6
3     Other
4     Other
5     Other
6     Other
7     Other
8     Other
9     Other
10    Other
11    Other
12    Other
13        6
14    Other
15    Other
16    Other
17    Other
18    Other
19    Other
20    Other
21    Other
22        6
23        6
24        6
25        6
26    Other
27    Other
28        6
29    Other
30    Other
31    Other
32    Other
dtype: object
```

Ejercicio 7: Write a Pandas program to calculate the frequency counts of each unique value of a given series.

In [12]:
```python
serie=pn.Series([4,5,6,7,9,5,11,0,12,34,3,4,5,6,2,1,3,4,3,4,3,5,6,6,6,6,7,45,6,7,4,1,1])

lista_unicos=[]
long=len(serie)

for i in serie:
    cont=0
    for j in serie:
        if i != j:
            cont+=1
    if cont == long-1: #El -1 es porque al revisar la serie entera se compara consigo mismo
        lista_unicos.append(int(i)/long)

print(lista_unicos)
```

```
[0.2727272727272727, 0.3333333333333333, 0.0, 0.36363636363636365, 1.0303030303030303, 0.06060606060606061, 1.3636363636363635]
```

Ejercicio 8: Write a Pandas program to get the first 3 rows of a given DataFrame.

In [17]:
```python
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

tabla=pn.DataFrame(exam_data,index=labels)
tabla

tabla.loc[['a','b','c']]
```

Out[17]:

	name	score	attempts	qualify
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes

Ejercicio 9: Write a Pandas program to count the number of rows and columns of a DataFrame.

In [21]:
```python
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

tabla=pn.DataFrame(exam_data,index=labels)

print ('El número de líneas es: ')
print(len(tabla['name']))

print()

print('El número de columnas es: ')
print(len(tabla.loc['a']))
```

```
El número de líneas es: 
10

El número de columnas es: 
4
------------
4
10
```

Ejercicio 10: Write a Pandas program to select the rows where the score is missing, i.e. is NaN

In [62]:
```python
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

tabla=pn.DataFrame(exam_data,index=labels)

NaN=np.nan
for i in labels:
    fila=tabla.loc[i]
    for j in fila.isnull():
        if j == True:
            print(fila)

tabla[tabla.isnull().any(axis=1)]
```

```
name        James
score         NaN
attempts        3
qualify        no
Name: d, dtype: object
name        Laura
score         NaN
attempts        1
qualify        no
Name: h, dtype: object
```

Out[62]:

	name	score	attempts	qualify
d	James	NaN	3	no
h	Laura	NaN	1	no

Ejercicio 11: Write a Pandas program to select the rows the score is between 15 and 20

In [74]:
```python
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

tabla=pn.DataFrame(exam_data,index=labels)

lista_indices=tabla.index
for i in lista_indices:
    linea=tabla.loc[i]
    if linea[1] >=15 and linea[1] <=20:
        print (linea)
        print ('\n')
```

```
name        Katherine
score            16.5
attempts            2
qualify           yes
Name: c, dtype: object


name        Michael
score            20
attempts          3
qualify         yes
Name: f, dtype: object


name        Jonas
score          19
attempts        1
qualify       yes
Name: j, dtype: object
```

Ejercicio 12: Write a Pandas program to change the score in row 'd' to 11.5

In [ ]:
```python
tabla.loc['d','score']=11.5
```

In [ ]:
```python
tabla
```

Ejercicio 13: Write a Pandas program to group by the first column and get second column as lists in rows.

In [ ]:
```
col1 col2
0 C1 1
1 C1 2
2 C2 3
3 C2 3
4 C2 4
5 C3 6
6 C2 5
```

In [75]:
```python
dic={'col1':['C1','C1','C2','C2','C2','C3','C2'],'col2':[1,2,3,3,4,6,5]}
tabla2=pn.DataFrame(dic,index=[0,1,2,3,4,5,6])
tabla2
```

Out[75]:

	col1	col2
0	C1	1
1	C1	2
2	C2	3
3	C2	3
4	C2	4
5	C3	6
6	C2	5

In [ ]: