

# Curso Python 2024

## Teoría-4: Introducción a Pandas



# Hoy:

- ✓ Repaso: Parámetros y argumentos
- ✓ Objeto Index
- ✓ Objeto Series
- ✓ Slicing .loc y .iloc
- ✓ Limpieza BBDD



# Repaso: Argumentos vs Parámetros

- **Parámetros**

Nombres de variables  
locales

## Tipos

posicionales

por defecto/opcionales

keyword

- **Argumentos**

Se pasan por referencia  
(memoria)

```
def my_fun(a, b):  
    # Some code  
    return
```

```
x1 = 100  
x2 = 'Hello'
```

```
my_fun(x1, x2)
```

# Repaso: Parámetros warning!

- Después de un parámetro por defecto, los siguientes deben ser por defecto.

```
def my_fun(a, b=5, c):  
    return a**2 + b**3 + c**4
```



my\_fun(2, 3) → ???

↓

a

# Repaso: Argumentos warning!

- Después de argumento keyword, los siguientes deben ser keyword.

```
def my_fun(a, b, c):  
    return a**2 + b**3 + c **4  
  
my_fun(b=1, 2, 3)
```

# \*args vs \*\*kwargs

- Agotar argumentos (exhaust)
- Captar en la variable el resto de argumentos posicionales o keyword

```
def my_fun(*args, **kwargs):  
    return args, kwargs
```

# Pandas: Introducción

- Librería de análisis de datos
- Por debajo (under the hood) trabaja con numpy



# Pandas: Introducción

- Para secuencias: indexing posicional (ej: listas)
- Pandas permite indexar mediante etiquetas (ej: diccionarios)

```
['a', 'b', 'c', 'd']  
0      1      2      3
```

```
{'a': 10, 'b': 20, 'c': 30}
```



# Pandas: pd.Index

- Se basa en numpy arrays
- Son arrays 1-D
- Secuencias: iterate, unpack, ...
- Varios tipos

```
pd.Index([1, 2, 3, 4, 5])
```



```
Index([1, 2, 3, 4, 5], dtype='int64')
```

# Pandas: pd.Index

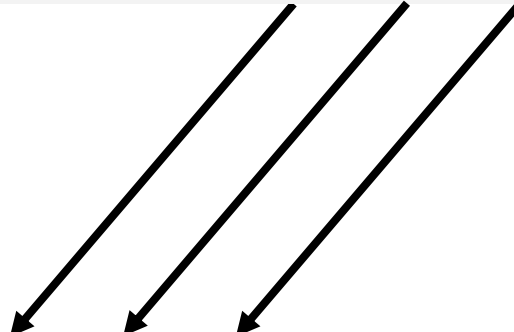
- Podemos realizar operaciones propias de conjuntos
- **&** (AND) Intersección
- **|** (OR) Unión
- **in** Pertenecer

```
pd.Index([1, 2, 3, 4, 5])
```

# Pandas: pd.Index

- Nos sirve para identificar elementos de una serie de objetos
- Si guarda **str** u otro el tipo de pd.Index será **object**

```
pd.Index([1, 2, 3, 4, 5])
```



```
Index([1, 2, 3, 4, 5], dtype='int64')
```

# Pandas: pd.Index

- Podemos tener valores repetidos

```
Index(['A', 'B', 'B', 'D'], dtype='object')
```

- Es distinto a las llaves de un diccionario

```
['a', 'b', 'c', 'd'] → ['b', 'c']
```

# Pandas: pd.Series

- Otro tipo de secuencias
- 1-D, numpy, ...
- Existen índices posicionales **implícitos**

```
a = pd.Series(['A', 'B', 'B', 'D'])
```

0	A
1	B
2	B
3	D

dtype: object

# Pandas: pd.Series

- Podemos usar un segundo tipo de índice
- Es lo que se llama índice **explícito**
- pd.Index es conveniente

```
a = pd.Series(['A', 'B', 'B', 'D'])
```

0	A	a
1	B	b
2	B	c
3	D	d

dtype: object

# Pandas: pd.Series

- Podemos indexar y utilizar slicing con ambos índices
- Pero para el índice explícito es distinto...

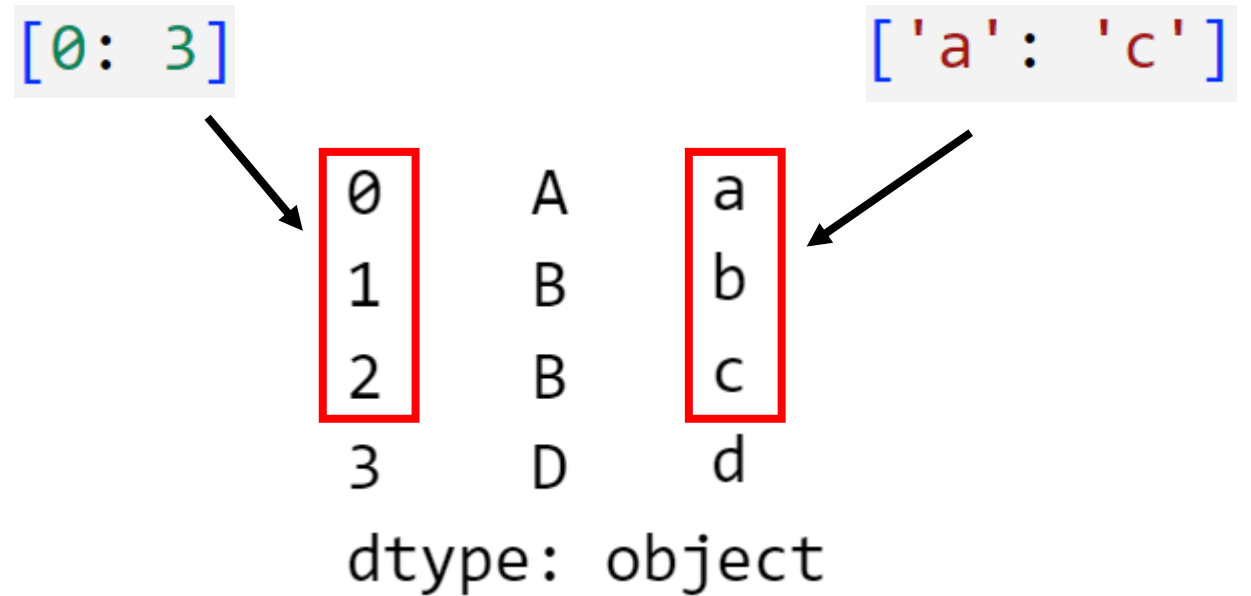
```
a = pd.Series(['A', 'B', 'B', 'D'])
```

0	A	a
1	B	b
2	B	c
3	D	d

dtype: object

# Pandas: pd.Series

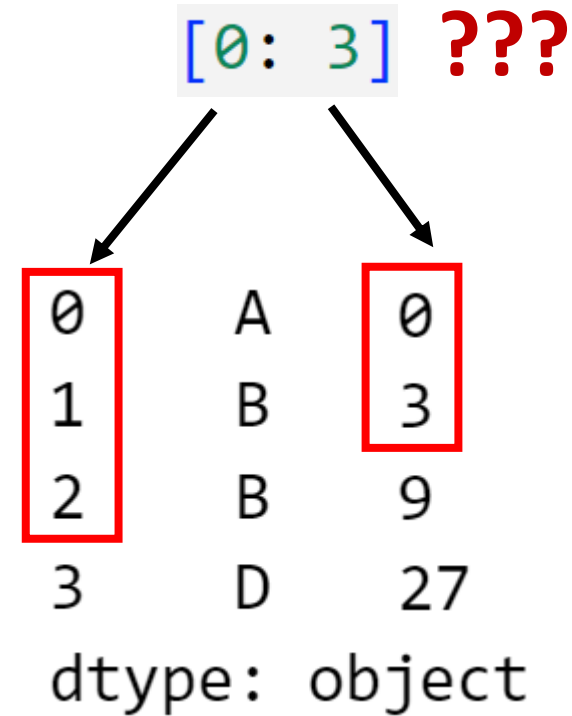
- Podemos indexar y utilizar slicing con ambos índices
- Pero para el índice explícito se incluye el límite derecho





# Pandas: pd.Series

- ¿Qué pasa si nuestros índices implícitos y explícitos ambos son enteros?
- Usamos **.loc** y **.iloc**



# Pandas: .loc y .iloc

- **.loc** para explícito
- **.iloc** para implícito

```
.iloc[0:3]
```



0  
1  
2

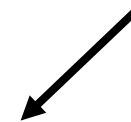
A

B

B

D

```
.loc[0:3]
```



0  
3

9

27

dtype: object