DMC | INSTITUTE

Módulo Python for Analytics – Sesión 03

Diploma Data Scientist

Docente: Geider Nuñuvero

REGLAS

Se requiere **puntualidad** para un mejor desarrollo del curso.

Para una mayor concentración **mantener silenciado el micrófono** durante la sesión.

Las preguntas se realizarán **a través del cha**t y en caso de que lo requieran **podrán activar el micrófono**.

Realizar las actividades y/o tareas encomendadas en **los plazos determinados**.

Identificarse en la sala Zoom con el primer nombre y primer apellido.

ITINERARIO

09:00 AM – 07:00 PM	***Soporte técnico DMC***
07:30 PM – 08:50 PM	***Agenda***
08:50 PM – 09:00 PM	***Pausa Activa***
09:00 PM – 10:30 PM	***Agenda***

Horario de Atención Área Académica y Soporte

Lunes a Viernes 09:00 am a 10:30 pm/ Sábados 09:00 am a 02:00 pm

Sesión 3

- Estructuras de control
- Librería Pandas: Series y Dataframes
- Principales métodos para Series y Dataframes
- Lectura y escritura desde varios formatos como txt, csv, xls.
- Seleccionando y filtrando Datasets
- Agrupando Datasets. Sentencias groupby, agg y pivot_table
- Combining and Merging Datasets

Estructuras de Control

Condicionales

Evalúa el **cumplimiento de una o más condiciones** para luego **ejecutar determinadas acciones**.

Estas sentencias son:
if , elif , else

Iterativas o bucles

Permite **ejecutar un mismo código** de manera repetida, **mientras se cumpla una condición**

Estas sentencias son:
for , while

Importante!
En Python es importante la identación(identation) o espaciado.

```
if password == "abcd1234":
    print("Access Granted")
else:
    print("Access Denied")
```

Condicionales (1/2)

if , else , elif

➔ Nos permiten evaluar **si una o más condiciones se cumplen**, para decir qué acción vamos a ejecutar.

➔ Las estructuras de control de flujo condicionales, se definen mediante el uso de tres palabras claves reservadas, del lenguaje: **if (si), else (sino) y elif (sino, si).**

➔ No es necesario la existencia del else ni elif.

```
if [condicion 1]:
    ...
elif [condicion 2]:
    ...
else:
    ...
```

Condicionales (2/2)

- Para describir la evaluación a realizar sobre una condición, se utilizan operadores relacionales (o de comparación): **==** **!=** **<=** **>=** **<** **>**
- Para evaluar más de una condición simultáneamente, es necesario utilizar alguno de los operadores lógicos: **and** **or** **xor**
- La condicional **if - else** también se puede escribir como una operación ternaria en una línea.

```
var = valor1 if [condicion] else valor2
```

Iterativas o Bucles (1/2)

while

- Se encarga de ejecutar una misma acción repetidamente **mientras que** una determinada condición se cumpla (sea **True**).

- Es importante que la condición cambie de estado en algún momento de ejecución, de los contrario tendrías un **bucle infinito.**

```
while [condicion]:
    ...
```

```
[4]  i = 0
     while i < 3:
         print(i)
         i += 1
```

```
0
1
2
```

Iterativas o Bucles (2/2)

for

- Se utiliza para **iterar sobre elementos** de una secuencia (string, tupla, lista) u otro objeto iterable.
- Si la secuencia está vacía, el bucle no se ejecuta ninguna vez.

```
for [iterator]:
    ...
```

```
[5]  for x in range(10):
         print('Hello!', x)
```

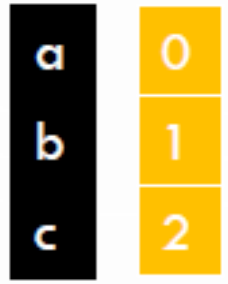
```
Hello! 0
Hello! 1
Hello! 2
Hello! 3
Hello! 4
Hello! 5
Hello! 6
Hello! 7
Hello! 8
Hello! 9
```

Series

- Los objetos series cuentan con métdoso para cálculo estadístico como mean, median,std..

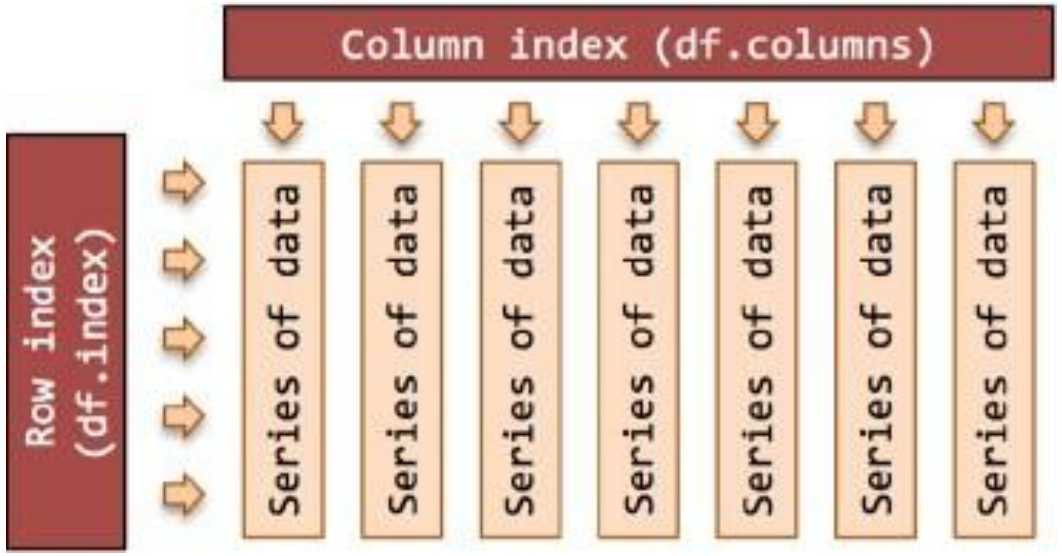
- El resultado de una operación aritmética (+,-,/,*,..):
  - Las series se alinean automáticamente según en sus etiquetas.
  - Cuando no hay alineación total, se obtiene la unión de todos los índices.
  - Se usa NaN para las etiquetas que no estén presente en todas las series.

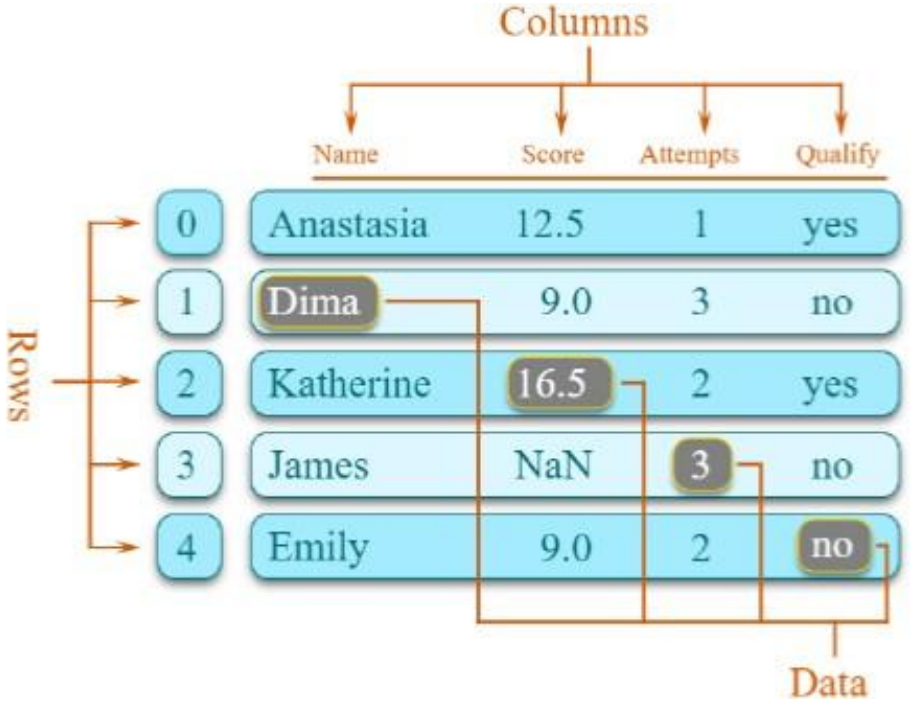
- Creación de una serie a partir de Numpy Array:
  - **pd.Series(np.arange(3))**
  - **pd.Series(np.arange(3), index=['a', 'b', 'c'])**



Data Frames

- El Data Frame permite almacenar y manipular datos tabulados en filas de observaciones y columnas de variables.





Métodos de Data Frame y Series

A diferencia de los atributos, los métodos de Python tienen paréntesis. Todos los atributos y métodos se pueden enumerar con un *dir()* function: `dir(df)`

df.method()	Description
head([n]), tail([n])	primera / última n filas
describe()	generar estadísticas descriptivas (solo para columnas numéricas)
max(), min()	devolver valores máximos / mínimos para todas las columnas numéricas
mean(), median()	devolver valores medios / medianos para todas las columnas numéricas
std()	Desviación Estándar
sample([n])	devuelve una muestra aleatoria del marco de datos
dropna()	descartar todos los registros con valores faltantes

Leer Datos en Pandas

```
#Read csv file
df = pd.read_csv("myfyle.csv",  sep =","  ,  encoding="utf-8")
```

Nota: El comando anterior tiene muchos argumentos opcionales para ajustar el proceso de importación de datos.

Hay varios comandos pandas para leer otros formatos de datos:

pd.read_excel('myfile.xlsx', sheet_name='Sheet1', na_values=['NA'])

pd.read_csv('myfile.txt' , sep='|', encoding = "ISO-8859-1")

pd.read_sas('myfile.sas7bdat')

pd.read_stata('myfile.dta')

Seleccionar columnas de Data Frame

- Subconjunto del marco de datos utilizando el nombre de la columna:

```
df ['sexo']
```

- Use el nombre de la columna como un atributo:

```
df.sex
```

Nota: hay un rango de atributo para los marcos de datos de pandas, por lo que para seleccionar una columna con un nombre de "rango" debemos usar el método 1

- Al seleccionar una columna, es posible usar un conjunto único de corchetes, pero el objeto resultante será una Serie (no un Data Frame):

```
df['salary']
```

- Cuando necesitamos seleccionar más de una columna y / o hacer que la salida sea un DataFrame, debemos usar corchetes dobles:

```
df[['rank','salary']]
```

Seleccionar filas de Data Frame

Si necesitamos seleccionar un rango de filas, podemos especificar el rango usando ":“

```
df[10:20]
```

Observe que la primera fila tiene una posición 0 y se omite el último valor en el rango: Entonces, para el rango 0:10, las primeras 10 filas se devuelven con las posiciones que comienzan con 0 y terminan con 9

Filtrando datos

Queremos seleccionar filtrando el data frame en el que básicamente la columna a sea mayor al valor de 5 y que la columna c sea mayor a 13

```
#Seleccion de un DataFrame usando criterios multiples de columnas
df[(df['a']>5) & (df['c'] > 13)]
```

Se puede usar cualquier operador booleano para subconjuntar los datos:
> mayor; > = mayor o igual;
<menos; <= menor o igual;
== igual; ! = no es igual;

```
#Selección para un DataFrame usando multiples columnas
newdf = df[(df['col1']>2) & (df['col2']==444)]
```

	col1	col2	col3
3	4	444.0	xyz

Método loc & iloc en Data Frame

Si necesitamos seleccionar un rango de filas, usando sus etiquetas podemos usar el método loc:

```
#Seleccionar los indices del 1 al 3
y las variables d y f :
df.loc[1:3,['d','f']]
```

	d	f
1	7	2019-03-01
2	11	2019-04-01
3	15	2019-05-01

Si necesitamos seleccionar un rango de filas y / o columnas, usando sus posiciones podemos usar el método iloc:

```
#Seleccionar los indices del 1 al 3 y
la posiciín de la variables 1, 3 y 5:
df.iloc[2:4,[1,3,5]]
```

	b	d	f
2	9	11	2019-04-01
3	13	15	2019-05-01

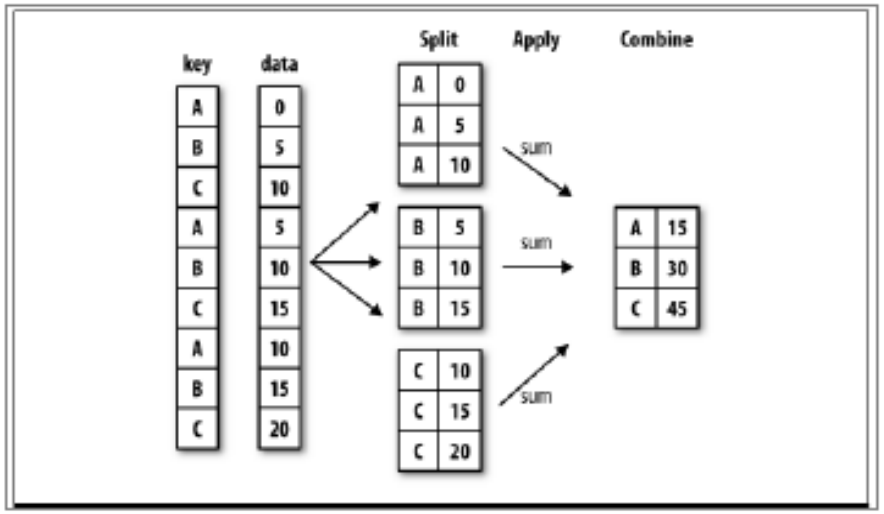
```
df.iloc[0]   # Primera fila del Dataframe
df.iloc[i]   #(i+1)th fila
df.iloc[-1]  # Ultima fila
```

```
df.iloc[:, 0]   # Primera columna
df.iloc[:, -1]  # Ultima columna
```

```
df.iloc[0:7]         # Primeras 7 final
df.iloc[:, 0:2]      # Primeras 2 columnas
df.iloc[1:3, 0:2]    # Segundo y tercera fila y las primeras 2 columnas
df.iloc[[0,5], [1,3]]   #1st y 6th fila y 2nd y 4th columnas
```

Data Frames: Realizando agrupaciones

- El método groupby permite hacer agregaciones de datos y, después, operaciones en base a esas agregaciones:
  - Sum.
  - Mean.
  - Median.
  - Std.
  - Size.
  - Describe
  - Quantile.



- El método agg permite hacer varios cálculos con los agregados.

`df.groupby(col)` - Returns a groupby object for values from one column

`df.groupby([col1,col2])` - Returns a groupby object values from multiple columns

`df.groupby(col1)[col2].mean()` - Returns the mean of the values in `col2`, grouped by the values in `col1` (mean can be replaced with almost any function from the statistics section)

`df.groupby(col1).agg(np.mean)` - Finds the average across all columns for every unique column 1 group

`df.apply(np.mean)` - Applies a function across each column

`df.apply(np.max, axis=1)` - Applies a function across each row

- El método agg permite hacer varios cálculos con los agregados.

In [75]:
```
# Base 2 de Perfil de Clientes
df_agre = dataset.groupby(['Departamento','Terminal','Color']).agg({
    'prom_llam_201908': ['count', 'min', 'max', 'mean', 'median', 'sum', 'std'],
    'prom_llam_201909': ['count', 'min', 'max', 'mean', 'median', 'sum', 'std']
}).reset_index()

df_agre.columns = [''.join(l).strip() for l in df_agre.columns]
df_agre.rename(columns=lambda x: '' + x if x != '' else x, inplace=True)
df_agre
```

Out[75]:

	Departamento	Terminal	Color	prom_llam_201908count	prom_llam_201908min	prom_llam_201908max	prom_llam_201908mean	prom_llam_20
0	AMAZONAS	ALCATEL 4003 PIXI3 4	NEGRO	4	0.0	0.6	0.150000	
1	AMAZONAS	BMOBILE AX675	NEGRO	1	0.0	0.0	0.000000	
2	AMAZONAS	BMOBILE K350	NEGRO	1	0.0	0.0	0.000000	
3	ANCASH	ALCATEL 1050	BLANCO	8	0.0	14.2	3.325000	
4	ANCASH	ALCATEL 1050	NEGRO	20	0.0	9.8	1.440000	
...	...	...	...	...	...	...	...	
		SONY						

- También permite aplicar cálculos específicos a cada grupo.

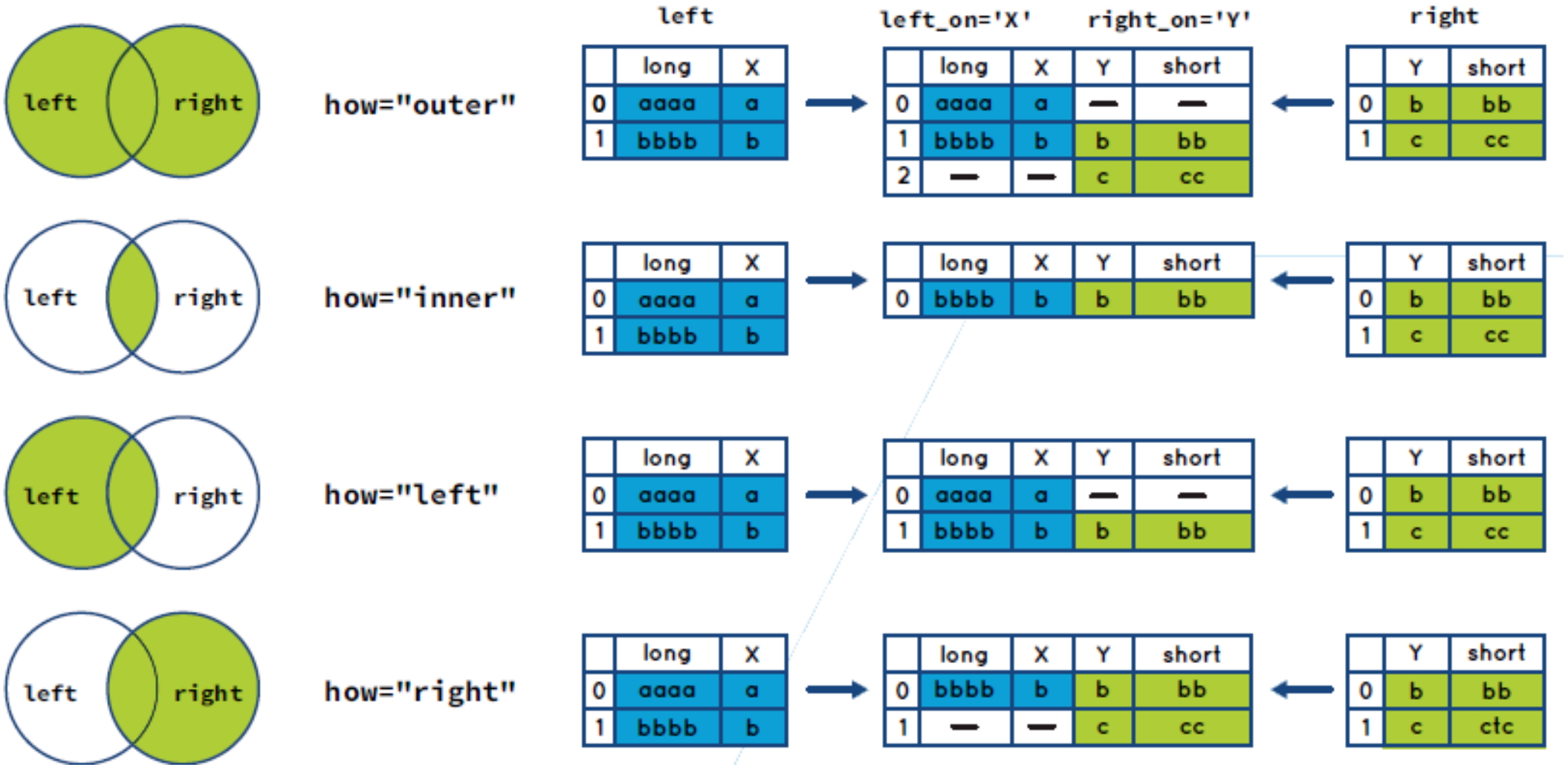
```
df_agre = dataset.query('Departamento  == "HUARAZ"')\
.groupby(['Terminal','Color']).agg({
    'prom_llam_201908': ['count', 'min', 'max', 'mean', 'median', 'sum', 'std'],
    'prom_llam_201909': ['count', 'min', 'max', 'mean', 'median', 'sum', 'std']
}).reset_index()

df_agre.columns = [''.join(l).strip() for l in df_agre.columns]
df_agre.rename(columns=lambda x: '' + x if x != 'codCliente' else x, inplace=True)
df_agre.head()
```

]:

	Terminal	Color	prom_llam_201908count	prom_llam_201908min	prom_llam_201908max	prom_llam_2019
0	ALCATEL 4003 PIXI3 4	BLANCO	1	5.6	5.6	
1	ALCATEL 4003 PIXI3 4	NEGRO	2	0.0	0.0	
2	ALCATEL 4027 PIXI3 4.5	BLANCO	2	0.6	10.2	
3	BMOBILE AX675	AZUL	1	0.0	0.0	
4	BMOBILE AX675	NEGRO	3	0.0	0.0	

Pandas: combinando Dataframes con merge



GRACIAS