

```
require( 'TEMPLATEPATH.DS.'yjscore/yjs_stylew.php');
$renderer
= $document->loadRenderer( 'module' );
$options
= array( 'style' => "raw" );
$module
= JModuleHelper::getModule( 'mod_menu' );
$stopmenu
= false; $subnav = false; $sidenav = false;
Main Menu
if ( $default_menu_style == 1 or $default_menu_style== 2 ) :
    $module->params = "menutype=$menu_name\nshowAllChildren=1\nclass_grow
    $stopmenu = $renderer->render( $module, $options );
    $menuclass = 'horiznav';
    $stopmenuclass = 'top_menu';
elseif ( $default_menu_style == 3 or $default_menu_style== 4 ) :
    $module->params = "menutype=$menu_name\nshowAllChildren=1\nclass_grow
    $stopmenu = $renderer->render( $module, $options );
    $menuclass = 'horiznav_d';
    $stopmenuclass = 'top_menu_d';
SPLIT MENU NO SUBS
elseif ( $default_menu_style == 5 ) :
    $module->params = "menutype=$menu_name\nstartLevel=$startLevel
    $stopmenu = $renderer->render( $module, $options );
    $menuclass = 'horiznav';
    $stopmenuclass = 'top_menu';
```



## Scientific Libraries 2: Pandas

Ing. Juan Camilo Correa Chica

## Scientific Libraries 2: Pandas

- ¿Qué es y para que sirve la librería Pandas?
- Instalación de Pandas
- Pandas DataFrame
- Funciones y operaciones sobre DataFrames con Pandas

## ¿Qué es Pandas?

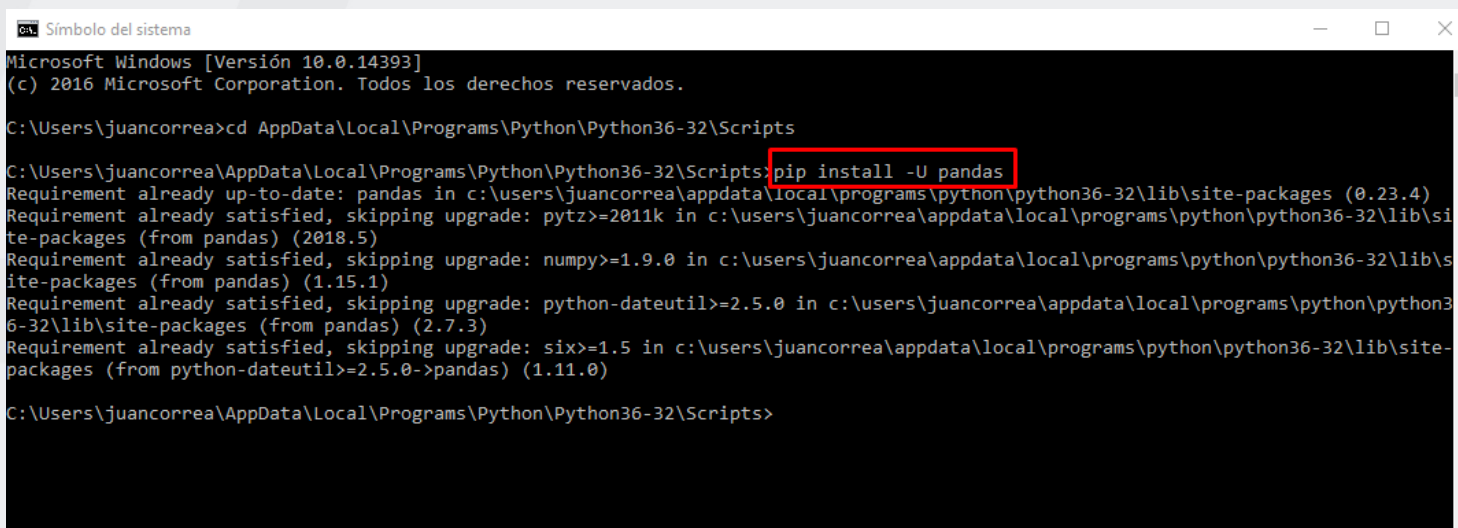
Pandas es una librería de Python construida a partir de las librerías Numpy y Matplotlib que se utiliza en el análisis de datos. Al igual que numpy trae consigo la estructura de datos “array”, pandas proporciona las siguientes estructuras útiles en el análisis de datos: Series, DataFrame, Panel, Panel4D y PanelND.

Pandas es especialmente útil en aplicaciones de BigData y de análisis de datos a partir de sistemas contruidos con base en técnicas de inteligencia artificial.

## Instalación de Pandas

Pandas se instala fácilmente en las más recientes versiones de Python con la ayuda del programa “pip”. En la carpeta raíz de la instalación de Python se debe buscar el ejecutable de pip y luego invocarlo para que instale la librería Pandas:

pip install -U pandas



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\juancorrea>cd AppData\Local\Programs\Python\Python36-32\Scripts

C:\Users\juancorrea\AppData\Local\Programs\Python\Python36-32\Scripts>pip install -U pandas
Requirement already up-to-date: pandas in c:\users\juancorrea\appdata\local\programs\python\python36-32\lib\site-packages (0.23.4)
Requirement already satisfied, skipping upgrade: pytz>=2011k in c:\users\juancorrea\appdata\local\programs\python\python36-32\lib\site-packages (from pandas) (2018.5)
Requirement already satisfied, skipping upgrade: numpy>=1.9.0 in c:\users\juancorrea\appdata\local\programs\python\python36-32\lib\site-packages (from pandas) (1.15.1)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.5.0 in c:\users\juancorrea\appdata\local\programs\python\python36-32\lib\site-packages (from pandas) (2.7.3)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\juancorrea\appdata\local\programs\python\python36-32\lib\site-packages (from python-dateutil>=2.5.0->pandas) (1.11.0)

C:\Users\juancorrea\AppData\Local\Programs\Python\Python36-32\Scripts>
```

## Series

Las Series son arreglos unidimensionales que pueden ser indexados a través de una etiqueta y no solamente por la posición del elemento en el arreglo. Se pueden generar a partir de diccionarios y de listas.

```
import pandas as pd

spanishPlayers = pd.Series( ['Casillas', 'Ramos', 'Pique', 'Puyol',
                             'Capdevila', 'Xabi Alonso', 'Busquets', 'Xavi Hernandez', 'Pedrito',
                             'Iniesta', 'Villa'], index=[1, 15, 3, 5, 11, 14, 16, 8, 18, 6, 7])

print "Spanish Football Players: \n%s" % spanishPlayers
```

## Data Frames

Un DataFrame es una estructura bidimensional que permite tabular los datos de forma similar a como se hace en bases de datos como SQL. A diferencia de los arreglos bidimensionales un DataFrame permite etiquetar las filas, que en pandas se conocen como índices, y las columnas de modo que la indexación se puede realizar como se hace con los diccionarios. Y además los elementos pueden ser de distinto tipo.

Columns

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasília	207847528

Index

A two-dimensional labeled data structure with columns of potentially different types

## Leyendo datos almacenados en archivos CSV

Para leer datos almacenados en archivos de valores separados por comas (csv) se utiliza la función de pandas `read_csv("ruta_archivo_csv")`

```
import pandas as pd

#Leyendo el archivo y almacenando los datos en formato de un DataFrame
custom_data_frame = pd.read_csv("arreglo_de_datos.csv")

#Visualizando el DataFrame completo
print(custom_data_frame)

#Visualizando las primeras N filas del DataFrame
print(custom_data_frame.head(N))

#Visualizando las ultimas N filas del DataFrame
print(custom_data_frame.tail(N))
```

## Métodos de los DataFrame

Distintas formas para construir un DataFrame (¡existen más!)

```
import pandas as pd
import numpy as np

#Creando un DataFrame a partir de un arreglo de numpy
dataframe1 = pd.DataFrame(np.array([[1,2,3,4], [5,6,7,8]]))

#A partir de un arreglo y especificando índices y columnas
aleatorio = np.random.random((5,5))
dataframe2 = pd.DataFrame(data = aleatorio, index = range(0,5),
                           columns = ['A', 'B', 'C', 'D', 'E'])

# A partir de un diccionario
dict = {'ID':[0,1,2,3,4], 'Nombre':['Alex', 'Diana', 'Sergio', 'Luisa', 'Sara'],
        'Ocupacion':['Estudiante', 'Programador', 'Estudiante', 'Ejecutivo', 'Archivista']}
dataframe3 = pd.DataFrame(data = dict)
```



## DataFrame vacío

Se puede construir un DataFrame vacío...

```
import pandas as pd
import numpy as np
```

```
#Creando un DataFrame vacío
```

```
dataframe_vacio = pd.DataFrame(np.nan, index = range(0,5),
                                columns = ['A', 'B', 'C', 'D', 'E'])
```

```
#Creando un DataFrame vacío
```

```
dataframe_vacio = pd.DataFrame(index = range(0,5),
                                columns = ['A', 'B', 'C', 'D', 'E'],
                                dtype = 'float')
```

## Obteniendo información básica de un DataFrame

```
import pandas as pd

#Dimensiones del DataFrame
m,n = data_frame.shape

#Indices del DataFrame
indx = list(data_frame.index)

#Columnas del DataFrame
cols = list(data_frame.columns)

#Informacion basica del DataFrame
info = data_frame.info()

#Cantidad de elementos con algun valor asignado (no nulos)
num = data_frame.count()

#Cantidad de indices
num_indx = len(data_frame.index)

#Cantidad de indices
num_cols = len(data_frame.columns)
```

## Operaciones fundamentales de los DataFrame

Seleccionar un índice o columna:

```
import pandas as pd

#Construyendo un DataFrame
data_frame = pd.DataFrame(np.array([[1,2,3,4], [5,6,7,8]]),
                           columns = ['A', 'B', 'C', 'D'])

#Seleccionando un índice
a = data_frame[0]
a = data_frame.ix[0]

#Seleccionando una columna (o varias)
x = data_frame['C']
x = data_frame.ix[:, ['C', 'D']]
```

Selección de elementos por posición y por label:

```
#Elementos por posicion
b = data_frame.iloc[0:1, :]
y = data_frame.iat[0, 3]

#Elementos por label
c = data_frame.loc[0:1, 'B']
w = data_frame.at[1, 'C']
```

## Operaciones fundamentales de los DataFrame

Agregar un índice o columna:

```
import pandas as pd

#Construyendo un DataFrame
data_frame = pd.DataFrame(np.array([[1,2,3,4], [5,6,7,8]]),
                           columns = ['A', 'B', 'C', 'D'])

#Agregando un índice (fila)
data_frame.ix[2]=[9,10,11,12]
data_frame.loc[3]=[12,14,15,16]

#Agregando una columna
data_frame['E']=[20,21,22,23]
data_frame.loc[:, 'F']=pd.Series([17,28,39,40], index=data_frame.index)
```

Eliminando un índice o una columna:

```
#Eliminando una fila
data_frame.drop(data_frame.index[3])

#Eliminando una columna
data_frame.drop('F', axis=1, inplace=True)
data_frame.drop(data_frame.columns[[4]], axis=1, inplace=True)
```

## Operaciones fundamentales de los DataFrame

Renombrar un índice o columna:

```
import pandas as pd

#Construyendo un DataFrame
data_frame = pd.DataFrame(np.array([[1,2,3,4], [5,6,7,8]]),
                           columns = ['A','B','C','D'])

newcols = { 'A':'columna_1', 'B':'columna_2', 'C':'columna_3',
            'D':'columna_4' }
newindex = {0: 'w', 1: 'x', 2: 'y', 3: 'z' }

#Renombrando las columnas
data_frame.rename(columns=newcols, inplace=True)

#Renombrando las filas
data_frame.rename(index=newindex, inplace=True)
```

## Operaciones fundamentales de los DataFrame

Reemplazar ocurrencias de elementos de tipo string:

```
import pandas as pd

#Construyendo un DataFrame
data_frame = pd.DataFrame(np.array([[ 'Bien', 'Mal', 'Excelente'], [ 'Mal', 'Excelente', 'Excelente'], [ 'Bien', 'Bien', 'Bien']]))

#Reemplazando los valores string por valores numericos
data_frame.replace([ 'Mal', 'Bien', 'Excelente'], [0, 1, 2], inplace=True)
```

Aplicar funciones a filas o columnas:

```
#Una funcion
def multbyt看wo(x): return x*2

#Aplicando función a una columna
j = data_frame['columna_2'].apply(multbyt看wo)

#Aplicando función a todos los elementos del dataframe
new_dataframe = data_frame.applymap(multbyt看wo)
```

## Operaciones fundamentales de los DataFrame

Reformando un DataFrame (reshaping):

Con la función pivot()

```
import pandas as pd

#Construyendo un DataFrame y reformándolo con pivot
data_frame = pd.DataFrame({'Empleado':['Juan', 'Mariana', 'Santiago', 'Karen'],
                           'Area':['Nomina', 'Sistemas', 'RH', 'Finanzas'], 'Puntuacion':[6.5, 7.8, 5.5, 8.8]})

reshaped_df = data_frame.pivot(index='Empleado', columns='Area', values='Puntuacion')
```

Con la función stack():

```
#Reformando con la función stack()
df = pd.DataFrame(data=[[0.34, 0.56],[0.25, 0.76],[0.63, 0.16]],index=range(1,4),
                  columns=[0,1])

dfx = df.stack()
```

## Operaciones fundamentales de los DataFrame

Reformando un DataFrame (reshaping):

Con la función melt()

```
import pandas as pd
```

```
#Construyendo un DataFrame y reformándolo con melt
```

```
data_frame = pd.DataFrame({'Area':['Potencia', 'Telecos', 'Control','Potencia',  
    'Telecos', 'Control','Potencia', 'Telecos', 'Control','Potencia', 'Telecos',  
    'Control'], 'Experimento':['a', 'b', 'b', 'd', 'c', 'a', 'a', 'd', 'b', 'd', 'c',  
    'a'], 'Resultado':[3.3, 4.7, 0.7, 6.6, 4.6, 2.8, 3.3, 9.2, 7.2, 1.4, 3.2, 6.9]})
```

```
df = pd.melt(data_frame, id_vars='Area', value_vars=['Experimento','Resultado'],  
value_name='Observaciones')
```



## Operaciones fundamentales de los DataFrame

Combinando y concatenando DataFrames:

Combinación con la función merge()

```
import pandas as pd

#Combiando dataframes con la función merge
df1 = pd.DataFrame({'vars':['a','b','c','d'], 'x':[1,2,3,4]})
df2 = pd.DataFrame({'vars':['a','f','c','e'], 'y':[5,6,7,8]})

df3=pd.merge(df1, df2, how='left', on='vars')
df4=pd.merge(df1, df2, how='right', on='vars')
df5=pd.merge(df1, df2, how='inner', on='vars')
df6=pd.merge(df1, df2, how='outer', on='vars')
```

Concatenación con la función join()

```
import pandas as pd

#Combiando dataframes con la función merge
df1 = pd.DataFrame({'vars':['a','b','c','d'], 'x':[1,2,3,4]})
df2 = pd.DataFrame({'vars':['a','f','c','e'], 'y':[5,6,7,8]})

df3 = df1.set_index('vars').join(df2.set_index('vars'))
```

## Operaciones fundamentales de los DataFrame

Combinando y concatenando DataFrames:

Concatenación con la función append()

```
import pandas as pd

#Concatenando una serie a un dataframe
s1 = pd.Series([-6, -12], index=range(0,2))
df = pd.DataFrame(data=[[1,2],[3,4],[5,6],[7,8]],index=range(0,4),
columns=[0,1])

df1=df.append(s1, ignore_index=True)

#Concatenando un dataframe a otro dataframe

df3=df.append(df1)
```

## Operaciones fundamentales de los DataFrame

Combinando y concatenando DataFrames:

Concatenación con la función concat()

```
import pandas as pd

#Concatenando un dataframe a otro dataframe

df1 = pd.DataFrame(data=[[1,2],[3,4],[5,6],[7,8]],index=range(0,4),
columns=[0,1])

df2 = pd.DataFrame(data=[[9,10],[11,12],[13,14],[15,15]],index=range(0,4),
columns=[0,1])

df3 = pd.concat([df1, df2], axis=1, join='inner') #Horizontal
df4 = pd.concat([df1, df2], axis=0, join='inner') #Vertical
```

## Operaciones fundamentales de los DataFrame

Iterando DataFrames:

```
import pandas as pd

#Iterando por filas y por columnas

df = pd.DataFrame(data=[[1,2],[3,4],[5,6],[7,8]],index=range(0,4),
columns=[0,1])

for index, row in df.iterrows() : #filas
    print(row[0], row[1])

for columns, col in df.iteritems() : #filas
    print(col[0], col[1])
```

## Otras funciones útiles....

```
import pandas as pd
```

```
#Agrupamiento
```

```
groupby()
```

```
#Reemplazar datos faltantes
```

```
dropna()
```

```
fillna()
```

```
#Manipular indices
```

```
set_index()
```

```
reset_index()
```

```
#Indexamiento avanzado
```

```
filter()
```

```
select()
```

```
query()
```

```
where()
```

```
#Ordenar y ranquear
```

```
sort_index()
```

```
sort_values()
```

```
rank()
```

## Otras funciones útiles....

```
import pandas as pd
```

```
#Operaciones aritmeticas
```

```
add()
```

```
sub()
```

```
div()
```

```
mul()
```

```
#Estadísticas
```

```
sum()
```

```
cumsum()
```

```
min()
```

```
max()
```

```
idxmin()
```

```
idxmax()
```

```
describe()
```

```
mean()
```

```
median()
```

```
std()
```

## Ejercicios prácticos utilizando pandas

- Lea en un dataframe el archivo Parks.csv y use la columna “park.key” como índice de las filas
- Obtenga las primeras 10 filas del dataframe
- Obtenga los valores de la fila ubicada en el índice 15 del dataframe
- Obtenga los valores de la fila indexada por “CLE04” en el dataframe
- Obtenga de manera simultánea los valores de las filas indexadas por “LBV01”, “MIA01”, “PHI10” y “SYR03”
- Obtenga los últimos 20 valores de la columna “park.alias”
- Implemente un nuevo dataframe con un subconjunto de las filas 10 a 19 del dataframe actual.
- Implemente un nuevo dataframe con un subconjunto de las filas 5 a 9 y las columnas “park.name” y “city” del dataframe actual.
- Obtenga los valores para la columna “city” cuyo campo “country” es “MX”
- Implemente un nuevo dataframe con un subconjunto de las filas cuya columna “state” tiene alguno de los siguientes valores: “OH”, “NY”, “CA” y “WI”

## Enlaces útiles Pandas

<https://pandas.pydata.org/pandas-docs/stable/>

<https://www.datacamp.com/community/tutorials/pandas-read-csv>

<https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python>

<https://www.datacamp.com/community/tutorials/recommender-systems-python>

<https://stackabuse.com/creating-a-simple-recommender-system-in-python-using-pandas/>

<https://datascienceplus.com/building-a-book-recommender-system-the-basics-knn-and-matrix-factorization/>

<https://blog.hacemoscontactos.com/2018/05/04/women-who-code-medellin-abril-2018-carga-de-datos-usando-python-pandas/>

<https://www.dataquest.io/blog/pandas-python-tutorial/>

<https://www.dataquest.io/blog/pandas-tutorial-python-2/>

<https://jarroba.com/pandas-python-ejemplos-parte-i-introduccion/>

<https://www.machinelearningplus.com/python/101-pandas-exercises-python/>

<https://www.hackerearth.com/practice/machine-learning/data-manipulation-visualisation-r-python/tutorial-data-manipulation-numpy-pandas-python/tutorial/>

<https://www.w3resource.com/python-exercises/pandas/index.php>