Curso *Ingeniería de Características*

Descargando datos

Julio Waissman Vilanova


Ejecuta en Google Colab

1. Descargando datos a la fuerza bruta

Vamos a ver primero como ir descargando datos y luego como lidiar con diferentes formatos. Es muy importante que, si los datos los vamos a cargar por única vez, descargar el conjunto de datos, tal como se encuentran, esto es `raw data`.

Vamos primero cargando las bibliotecas necesarias:

```
import os  # Para manejo de archivos y directorios
import urllib.request # Una forma estandard de descargar datos
# import requests # Otra forma no de las librerías de uso comun

import datetime # Fecha de descarga
import pandas as pd # Solo para ver el archivo descargado
import zipfile # Descompresión de archivos
```

Es importante saber en donde nos encontramos y crear los subdirectorios necesarios para guardar los datos de manera ordenada. Tambien es importante evitar cargar datos que ya han sido descargados anteriormente.

```
# pwd
print(os.getcwd())

#  Estos son los datos que vamos a descargar y donde vamos a guardarlos
desaparecidos_RNPDNO_url = "http://www.datamx.io/dataset/fdd2ca20-ee70-4a31-9b
```

```
desaparecidos_RNPDNO_archivo = "desaparecidosRNPDNO.zip"
desaparecidos_corte_nacional_url = "http://www.datamx.io/dataset/fdd2ca20-ee70
desaparecidos_corte_nacional_archivo = "desaparecidos_nacional.json"
subdir = "./data/"
```

```
/content
```

```
if not os.path.exists(desaparecidos_RNPDNO_archivo):
    if not os.path.exists(subdir):
        os.makedirs(subdir)
    urllib.request.urlretrieve(desaparecidos_RNPDNO_url, subdir + desaparecido
    with zipfile.ZipFile(subdir + desaparecidos_RNPDNO_archivo, "r") as zip_re
        zip_ref.extractall(subdir)

    urllib.request.urlretrieve(desaparecidos_corte_nacional_url, subdir + desa

    with open(subdir + "info.txt", 'w') as f:
        f.write("Archivos sobre personas desaparecidas\n")
        info = """
        Datos de desaparecidos, corte nacional y desagregación a nivel estatal
        por edad, por sexo, por nacionalidad, por año de desaparición y por me
        de desaparición para los últimos 12 meses.

        Los datos se obtuvieron del RNPDNO con fecha de 03 de agosto de 2021
        (la base de datos no se ha actualizado últimamente)

        """
        f.write(info + '\n')
        f.write("Descargado el " + datetime.datetime.now().strftime("%Y-%m-%d
        f.write("Desde: " + desaparecidos_RNPDNO_url + "\n")
        f.write("Nombre: " + desaparecidos_RNPDNO_archivo + "\n")
        f.write("Agregados nacionales descargados desde: " + desaparecidos_cor
        f.write("Nombre: " + desaparecidos_corte_nacional_archivo + "\n")
```

2. Archivos en formato `json`

Los archivos en formato json son posiblemente los más utilizados actualmente para transferir información por internet, ya que se usa en prácticamente todas las REST API. Como acabamos de ver es normal tener que enfrentarse con archivos `json` pésimamente o nada documentados, por lo que es necesario saber como tratarlos.

Vamos a ver como se hace eso utilizando la bibloteca de `json` y la de `pandas`. Para `pandas` les recomiendo, si no lo conocen, de darle una vuelta a la documentación y los tutoriales que está muy bien hecha. O a el curso básico de Kaggle.

Sobre `json`, posiblemente la página con la especificación sea más que suficiente.

Vamos a hacer un ejemplito sencillo y carismático revisando los repositorios de github y les voy a dejar que exploren los `json` de los archivos de personas desaparecidas.

```
import pandas as pd # Esto es como una segunda piel
import json # Una forma estandar de leer archivos json

archivo_url = "https://api.github.com/users/google/repos"
archivo_nombre = "repos-google.json"
subdir = "./data/"

if not os.path.exists(subdir + archivo_nombre):
    if not os.path.exists(subdir):
        os.makedirs(subdir)
    urllib.request.urlretrieve(archivo_url, subdir + archivo_nombre)
```

Vamos primero a ver como le hacemos con `pandas`

```
df_repos = pd.read_json(subdir + archivo_nombre)

df_repos.head()
```

	id	node_id	name	full_name
0	460600860	R_kgDOG3Q2HA	.allstar	google/.allstar
1	170908616	MDEwOlJlcG9zaXRvcnkxNzA5MDg2MTY=	.github	google/.github
2	143044068	MDEwOlJlcG9zaXRvcnkxNDMwNDQwNjg=	0x0g-2018-badge	google/0x0g-2018-badge
3	424674738	R_kgDOGVAFsg	aarch64-esr-decoder	google/aarch64-esr-decoder
4	487987687	R_kgDOHRYZ5w	aarch64-paging	google/aarch64-paging

5 rows × 79 columns

```
df_repos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 79 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      30 non-null     int64
 1   node_id                 30 non-null     object
 2   name                    30 non-null     object
 3   full_name               30 non-null     object
 4   private                 30 non-null     bool
 5   owner                   30 non-null     object
 6   html_url                30 non-null     object
 7   description             13 non-null     object
 8   fork                    30 non-null     bool
 9   url                     30 non-null     object
 10  forks_url               30 non-null     object
 11  keys_url                30 non-null     object
 12  collaborators_url       30 non-null     object
 13  teams_url               30 non-null     object
 14  hooks_url               30 non-null     object
 15  issue_events_url        30 non-null     object
 16  events_url              30 non-null     object
 17  assignees_url           30 non-null     object
 18  branches_url            30 non-null     object
 19  tags_url                30 non-null     object
 20  blobs_url               30 non-null     object
 21  git_tags_url            30 non-null     object
 22  git_refs_url            30 non-null     object
 23  trees_url               30 non-null     object
 24  statuses_url            30 non-null     object
 25  languages_url           30 non-null     object
 26  stargazers_url          30 non-null     object
 27  contributors_url        30 non-null     object
 28  subscribers_url         30 non-null     object
 29  subscription_url        30 non-null     object
 30  commits_url             30 non-null     object
 31  git_commits_url         30 non-null     object
 32  comments_url            30 non-null     object
 33  issue_comment_url       30 non-null     object
 34  contents_url            30 non-null     object
 35  compare_url             30 non-null     object
 36  merges_url              30 non-null     object
 37  archive_url             30 non-null     object
 38  downloads_url           30 non-null     object
 39  issues_url              30 non-null     object
 40  pulls_url               30 non-null     object
 41  milestones_url          30 non-null     object
 42  notifications_url       30 non-null     object
 43  labels_url              30 non-null     object
 44  releases_url            30 non-null     object
 45  deployments_url         30 non-null     object
 46  created_at              30 non-null     datetime64[ns, UTC]
 47  updated_at              30 non-null     datetime64[ns, UTC]
```

```
 48  pushed_at                       30 non-null     datetime64[ns, UTC]
 49  git_url                         30 non-null     object
 50  ssh_url                         30 non-null     object
 51  clone_url                       30 non-null     object
 52  svn_url                         30 non-null     object
```

y ahora como le hacemos con la biblioteca de `json`

```
with open(subdir + archivo_nombre, 'r') as fp:
    repos = json.load(fp)

print(f"\nNúmero de entradas: {len(repos)}")
print(f"\nNombre de los atributos: { ', '.join(repos[0].keys())}")
print(f"\nAtributos de 'owner': {', '.join(repos[0]['owner'].keys())}")
```

```
Número de entradas: 30

Nombre de los atributos: id, node_id, name, full_name, private, owner, htm

Atributos de 'owner': login, id, node_id, avatar_url, gravatar_id, url, htt
```

Ejercicio

Utiliza los archivos `json` descargados con el detalle a nivel estatal, y genera unos 3 `DataFrame` con información sobre personas desaparecidas dependiendo de diferentes características.

```
# 1er DataFrame: Cantidad de Desaparecidos por Estado.

# Podemos ver que el primer JSON de la categoría espacial se trata de los tota
# de desaparecidos, por sexo y por estado.

with open("data/datos_procesados/estados/0.json") as f:
  desap_json = json.load(f)

desap_df = pd.read_json(json.dumps(desap_json["espacial"], indent=4))

desap_df
```

```
<ipython-input-174-6bf55edac8c2>:9: FutureWarning: Passing literal json to
  desap_df = pd.read_json(json.dumps(desap_json["espacial"], indent=4))
```

	Hombre	Indeterminado	Mujer
AGUASCALIENTES	1848	3	2537
BAJA CALIFORNIA	2065	5	1938
BAJA CALIFORNIA SUR	582	2	214
CAMPECHE	243	0	504
CHIAPAS	1344	0	2060

CHIAPAS	1544	0	2000
CHIHUAHUA	8012	4	4306
CIUDAD DE MEXICO	5628	167	4695
COAHUILA	2964	2	1054
COLIMA	2047	2	1710
DURANGO	1610	1	1220
ESTADO DE MEXICO	18070	17	21669
GUANAJUATO	7118	228	7744
GUERRERO	3663	8	1436
HIDALGO	1264	5	1614
JALISCO	15845	16	4582
MICHOACAN	4664	7	1899
MORELOS	1557	2	1611
NAYARIT	1221	4	315
NUEVO LEON	5600	1	3413
OAXACA	1465	8	2150
PUEBLA	4328	5	5020
QUERETARO	1135	1	1084
QUINTANA ROO	880	0	1187
SAN LUIS POTOSI	654	2	636
SE DESCONOCE	625	74	293
SINALOA	8184	4	2466
SONORA	4345	7	2223
TABASCO	319	1	499
TAMAULIPAS	11624	19	5387
TLAXCALA	116	1	114
VERACRUZ	4719	17	2075
YUCATAN	2622	1	4209
ZACATECAS	2263	0	978

Next steps: Generate code with `desap_df` View recommended plots New interactive sheet

```
# Veamos entonces los totales para solo el estado de Sonora.
```

```
desap_df.loc["SONORA"]
```

	SONORA
Hombre	4345
Indeterminado	7
Mujer	2223

dtype: int64

```
# Podemos encontrar los totales más delimitados por municipios del estado de S
# en el JSON 26.

with open("data/datos_procesados/estados/26.json") as f:
  desap_json = json.load(f)

desap_sonora_df = pd.read_json(json.dumps(desap_json["espacial"], indent=4))

desap_sonora_df
```

```
<ipython-input-176-47549f6b511f>:7: FutureWarning: Passing literal json to
  desap_sonora_df = pd.read_json(json.dumps(desap_json["espacial"], indent=
```

	Hombres	Indeterminado	Mujeres
ACONCHI	3	0	3
AGUA PRIETA	296	0	174
ALAMOS	28	0	12
ALTAR	54	0	12
ARIVECHI	1	0	0
...	...	...	...
TUBUTAMA	0	0	1
URES	5	0	1
VILLA HIDALGO	1	0	0
VILLA PESQUEIRA	0	0	1
YÉCORA	4	0	2

64 rows × 3 columns

Next steps: Generate code with desap_sonora_df | recommended | New interactive sheet

```
# Delimitemos ahora a los totales de solo el municipio de Hermosillo.

desap_sonora_df.loc["HERMOSILLO"]
```

	HERMOSILLO
Hombres	970
Indeterminado	0
Mujeres	644

dtype: int64

```
# Finalmente, veamos en qué posición se encuentra Hermosillo en cuanto a canti
# de desaparecidos, comparado con los demás municipios.

#HOMBRES
desap_hombres_hmo = desap_sonora_df.sort_values(by='Hombres', ascending=False,
desap_hombres_hmo['pos'] = range(1, len(desap_hombres_hmo) + 1)

hombres_pos_hmo = desap_hombres_hmo.loc["HERMOSILLO"].loc["pos"]

#MUJERES
desap_mujeres_hmo = desap_sonora_df.sort_values(by='Mujeres', ascending=False,
desap_mujeres_hmo['pos'] = range(1, len(desap_mujeres_hmo) + 1)

mujeres_pos_hmo = desap_mujeres_hmo.loc["HERMOSILLO"].loc["pos"]

#INDETERMINADOS
desap_indeterminados_hmo = desap_sonora_df.sort_values(by='Indeterminado', asc
desap_indeterminados_hmo['pos'] = range(1, len(desap_indeterminados_hmo) + 1)

indeterminados_pos_hmo = desap_indeterminados_hmo.loc["HERMOSILLO"].loc["pos"]

print("Lugar de Hombres Desaparecidos:\t" + str(hombres_pos_hmo))
print("Lugar de Mujeres Desaparecidas:\t" + str(mujeres_pos_hmo))
print("Lugar de Indeterminados Desaparecidos:\t" + str(indeterminados_pos_hmo)
```

```
Lugar de Hombres Desaparecidos: 1
Lugar de Mujeres Desaparecidas: 1
Lugar de Indeterminados Desaparecidos:  61
```

Ya que Hermosillo es el municpio con mayor población, los primeros dos lugares son de esperarse. Sin embargo, es notorio que caiga tan abajo en su posición de desaparecidos de sexo indeterminado. Sería interesanteun análisis de ello.

```
# 2do DataFrame: Periodistas Desaparecidos

# Podemos también ver cuantos periodistas desaparecieron en México. Este conju
# de datos no contiene muchos valores, lo cual se ejemplifica con los periodis
# desaparecidos mensualmente por año.

with open("data/datos_procesados/por_categoria/periodista.json") as f:
  periodistas_json = json.load(f)

periodistas_df = pd.read_json(json.dumps(periodistas_json["mensual_ultimo_anio
```

```
periodistas_df
```

```
<ipython-input-179-c3a10082b0d3>:10: FutureWarning: Passing literal json t
  periodistas_df = pd.read_json(json.dumps(periodistas_json["mensual_ultim
```

	Hombre	Indeterminado	Mujer
2012-NOVIEMBRE	1	0	0
2020-FEBRERO	1	0	0
2020-NOVIEMBRE	0	0	1
2021-MARZO	2	0	0

Next steps: Generate code with periodistas_df recommended New interactive sheet

```
# Podemos ver que, en total, solo hubo 5 periodistas desaparecidos registrados
# en los años 2012, 2020 y 2021.

# Veamos sus edades.

periodistas_df = pd.read_json(json.dumps(periodistas_json["por_edad"], indent=

periodistas_df
```

```
<ipython-input-180-623210266abf>:6: FutureWarning: Passing literal json to
  periodistas_df = pd.read_json(json.dumps(periodistas_json["por_edad"], i
<ipython-input-180-623210266abf>:6: FutureWarning: The behavior of 'to_dat
  periodistas_df = pd.read_json(json.dumps(periodistas_json["por_edad"], i
<ipython-input-180-623210266abf>:6: FutureWarning: The behavior of 'to_dat
  periodistas_df = pd.read_json(json.dumps(periodistas_json["por_edad"], i
<ipython-input-180-623210266abf>:6: FutureWarning: The behavior of 'to_dat
  periodistas_df = pd.read_json(json.dumps(periodistas_json["por_edad"], i
```

	Hombres	Indeterminado	Mujeres
15	0	0	1
24	1	0	0
33	1	0	0
37	1	0	0
SIN EDAD DE REFERENCIA	1	0	0

Next steps: Generate code with periodistas_df recommended New interactive sheet

Debido a la poca cantidad de datos, no es posible obtener una conclusión con respecto a los periodistas desaparecidos.

```
# 3er DataFrame: Desaparecidos por Edades en Sonora

# Al buscar a los periodistas desaparecidos, fue interesante ver un conteo por
# edades. Por lo tanto, hagamos eso mismo pero para el conteo que teníamos en
# el estado de Sonora.

with open("data/datos_procesados/estados/26.json") as f:
  edades_json = json.load(f)

edades_df = pd.read_json(json.dumps(edades_json["por_edad"], indent=4))

edades_df
```

```
<ipython-input-181-d8433e3be8af>:10: FutureWarning: Passing literal json t
  edades_df = pd.read_json(json.dumps(edades_json["por_edad"], indent=4))
<ipython-input-181-d8433e3be8af>:10: FutureWarning: The behavior of 'to_da
  edades_df = pd.read_json(json.dumps(edades_json["por_edad"], indent=4))
<ipython-input-181-d8433e3be8af>:10: FutureWarning: The behavior of 'to_da
  edades_df = pd.read_json(json.dumps(edades_json["por_edad"], indent=4))
<ipython-input-181-d8433e3be8af>:10: FutureWarning: The behavior of 'to_da
  edades_df = pd.read_json(json.dumps(edades_json["por_edad"], indent=4))
```

	Hombres	Indeterminado	Mujeres
0	73	0	73
1	4	0	7
10	14	0	4
11	8	0	10
12	27	0	43
...	...	...	...
9	9	0	3
91	1	0	0
92	2	0	0
95	1	0	0
SIN EDAD DE REFERENCIA	757	2	485

91 rows × 3 columns

Next steps: Generate code with edades_df | View recommended plots | New interactive sheet

```
# Veamos qué edades son las que más desaparecidos tienen para hombres, mujeres
# indeterminados.

#HOMBRES
desap_hombres_hmo = edades_df.sort_values(by='Hombres', ascending=False, na_po
desap_hombres_hmo['pos'] = range(1, len(desap_hombres_hmo) + 1)
```

```
desap_hombres_hmo[["Hombres", "pos"]]
```

	Hombres	pos
SIN EDAD DE REFERENCIA	757	1
25	117	2
32	114	3
30	109	4
29	107	5
...	...	...
84	1	87
70	1	88
91	1	89
95	1	90
83	1	91

91 rows × 2 columns

```
# Podemos ver que el primer lugar es para la gran cantidad de desaparecidos pa
# quienes no se cuenta con registro de edad. Sin embargo, desde el 2do lugar,
# el 5to lugar, son solo edades entre 25 y 32.

# Ahora, veamos las edades para las mujeres.

#MUJERES
desap_mujeres_hmo = edades_df.sort_values(by='Mujeres', ascending=False, na_po
desap_mujeres_hmo['pos'] = range(1, len(desap_mujeres_hmo) + 1)

desap_mujeres_hmo[["Mujeres", "pos"]]
```

	Mujeres	pos
SIN EDAD DE REFERENCIA	485	1
14	221	2
15	209	3
16	166	4
17	120	5
...	...	...
69	0	87
78	0	88
77	0	89
70	0	90

82	0	91

91 rows × 2 columns

```
# Al igual que en el caso de los hombres, el 1er lugar son las personas para q
# no hay edad de referencia. Sin embargo, del 2do al 5to lugar tenemos solo ed
# de entre 14 a 17 años.

# Finalmente, veamos para las personas de sexo indeterminado.

#INDETERMINADO
desap_indeterminado_hmo = edades_df.sort_values(by='Indeterminado', ascending=
desap_indeterminado_hmo['pos'] = range(1, len(desap_indeterminado_hmo) + 1)

desap_indeterminado_hmo[["Indeterminado", "pos"]]
```

	Indeterminado	pos
SIN EDAD DE REFERENCIA	2	1
25	2	2
54	1	3
27	1	4
31	1	5
...	...	...
36	0	87
35	0	88
34	0	89
33	0	90
5	0	91

91 rows × 2 columns

Aquí aunque tengamos de nuevo como 1er lugar a las personas para quienes no tenemos edad de referencia, el resto de lugares no parecen estar en un rango de edades especifico que nos haga notar algo en particular.

3. Archivos xml

Los archivos *xml* son una manera de compartir información a través de internet o de guardar información con formatos genéricos que sigue siendo muy utilizada hoy en día. En general lidiar con archivos xml es una pesadilla y se necesita explorarlos con calma y revisarlos bien antes de usarlos.

La definición del formato y su uso se puede revisar en [este tutorial de la w3schools](). Vamos a ver un ejemplo sencillo basado en la librería [xml.etree.ElementTree]() que viene de base en python:

```
import xml.etree.ElementTree as et

archivo_url = "https://github.com/mcd-unison/ing-caract/raw/main/ejemplos/inte
archivo_nombre = "ejemplito.xml"
subdir = "./data/"

if not os.path.exists(subdir + archivo_nombre):
    if not os.path.exists(subdir):
        os.makedirs(subdir)
    urllib.request.urlretrieve(archivo_url, subdir + archivo_nombre)


desayunos = et.parse(subdir + archivo_nombre)

for (i, des) in enumerate(desayunos.getroot()):
    print("Opción {}:".format(i+1))
    for prop in des:
        print("\t{}: {}".format(prop.tag, prop.text.strip()))

# Se puede buscar por etiquetas y subetiquetas

print("Los desayunos disponibles son: " +
      ", ".join([p.text for p in desayunos.findall("food/name")]))

# ¿Como se podría poner esta información en un DataFrame de `pandas`?
# Agreguen tanto código como consideren necesario.
```

```
    Opción 1:
            name: Belgian Waffles
            price: $5.95
            description: Two of our famous Belgian Waffles with plenty of real
            calories: 650
    Opción 2:
            name: Strawberry Belgian Waffles
            price: $7.95
            description: Light Belgian waffles covered with strawberries and wl
            calories: 900
    Opción 3:
            name: Berry-Berry Belgian Waffles
            price: $8.95
            description: Belgian waffles covered with assorted fresh berries ar
            calories: 900
    Opción 4:
            name: French Toast
            price: $4.50
            description: Thick slices made from our homemade sourdough bread
            calories: 600
    Opción 5:
            name: Homestyle Breakfast
            price: $6.95
            description: Two eggs, bacon or sausage, toast, and our ever-popul
```

```
            description: Two eggs, bacon or sausage, toast, and our ever-popul
            calories: 950
    Los desayunos disponibles son: Belgian Waffles, Strawberry Belgian Waffles
```

Wikipedia es un buen ejemplo de un lugar donde la información se guarda y se descarga en forma de archivos xml. Por ejemplo, si queremos descargar datos de la wikipedia con su herramienta de exportación en python utilizando las categorias definidas por Wikipedia. Para hacerlo en forma programática es ecesario usar la API de Mediawiki que veremos más adelante.

Por el momento descargemos unos datos de *wikipedia* y hagamos el ejercicio de tratar de entender la estructura del árbol.

```
archivo_url = "https://github.com/mcd-unison/ing-caract/raw/main/ejemplos/inte
archivo_nombre = "poetas.xml"
subdir = "./data/"

if not os.path.exists(subdir + archivo_nombre):
    if not os.path.exists(subdir):
        os.makedirs(subdir)
    urllib.request.urlretrieve(archivo_url, subdir + archivo_nombre)


poetas = et.parse(subdir + archivo_nombre)
```

Ejercicio

Entender la estructura del archivo `xml` de poetas y generar un `DataFrame` con la información más importante. No olvides de comentar tu código y explicar la estructura del archivo `xml`

```
# Primero, podemos hacer un desglose de todos los poestas que tenemos de una m
# similar al ejemplo anterior. Esto solo como un comienzo para tratar de darno
# una idea de su estructura.

for (i, des) in enumerate(poetas.getroot()):
    print("Poeta {}:".format(i+1))
    for prop in des:
        print("\t {} : {}".format(prop.tag, prop.text.strip()))
```

```
    Poeta 1:
            {http://www.mediawiki.org/xml/export-0.10/}sitename : Wikipedia
            {http://www.mediawiki.org/xml/export-0.10/}dbname : eswiki
            {http://www.mediawiki.org/xml/export-0.10/}base : https://es.wiki
            {http://www.mediawiki.org/xml/export-0.10/}generator : MediaWiki
            {http://www.mediawiki.org/xml/export-0.10/}case : first-letter
            {http://www.mediawiki.org/xml/export-0.10/}namespaces :
    Poeta 2:
            {http://www.mediawiki.org/xml/export-0.10/}title : Julia Morilla
```

```
                {http://www.mediawiki.org/xml/export-0.10/}title : Julia Moritta (
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 4949229
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 3:
                {http://www.mediawiki.org/xml/export-0.10/}title : Luis Negreti
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 5105749
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 4:
                {http://www.mediawiki.org/xml/export-0.10/}title : Poldy Bird
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 4477192
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 5:
                {http://www.mediawiki.org/xml/export-0.10/}title : Ana María Shua
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 423422
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 6:
                {http://www.mediawiki.org/xml/export-0.10/}title : León Benarós
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 4284479
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 7:
                {http://www.mediawiki.org/xml/export-0.10/}title : Alejandro Gonza
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 5436786
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 8:
                {http://www.mediawiki.org/xml/export-0.10/}title : Silvia Schujer
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 3148146
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 9:
                {http://www.mediawiki.org/xml/export-0.10/}title : Laura Devetach
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 2428185
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 10:
                {http://www.mediawiki.org/xml/export-0.10/}title : Graciela Repún
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 1108758
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 11:
                {http://www.mediawiki.org/xml/export-0.10/}title : María Cristina
                {http://www.mediawiki.org/xml/export-0.10/}ns : 0
                {http://www.mediawiki.org/xml/export-0.10/}id : 3222523
                {http://www.mediawiki.org/xml/export-0.10/}revision :
        Poeta 12:
```

```
# Se trata entonces de una lista de poetas con su ID y todo un objeto de revis
# Podemos crear un DataFrame sencillo (ya que se trata de un ejercicio de ejem
# para tener la información de cada poeta organizada.

# Primero importamos Numpy
import numpy as np

# Ahora, creamos un array vacío donde ir guardando los datos que saquemos de c
```

```
# Ahora, creamos un array vacío donde iré guardando los datos que saquemos de c
data = []

# Ahora, iteramos sobre todos las subestructuras, las cuales ya sabemos son, c
# todas, los poetas. Solo la primera tiene información sobre la pagina en sí.
for child in poetas.getroot():
  # Si la subestructura es "page", sabemos que tiene información de un poeta.
  if (child.tag == "{http://www.mediawiki.org/xml/export-0.10/}page"):
    # Primero, generamos un array para guardar el renglon de datos de un poeta
    row = []

    # Luego, obtenemos la estructura del titulo, la cual es subestructura de "
    title = child.find("{http://www.mediawiki.org/xml/export-0.10/}title")
    # Agregamos el titulo como un dato al renglon.
    row.append(title.text.strip())

    # Hacemos exactamente lo mismo para el ID, ya que también es subestructura
    id = child.find("{http://www.mediawiki.org/xml/export-0.10/}id")
    row.append(id.text.strip())

    # Para el texto de la revisión es ligeramente distinto. Comenzamos obtenie
    # la subestructura "revision".
    revision_parent = child.find("{http://www.mediawiki.org/xml/export-0.10/}r
    # El texto se encuentra en realidad en una estructura un nivel más abajo,
    # "text". Sacamos ese texto y lo agregamos al renglon.
    revision_text = revision_parent.find("{http://www.mediawiki.org/xml/export
    row.append(revision_text.text.strip())

    # Ahora, agregamos ese renglon al array de datos.
    data.append(row)

# Convertimos el array a un array de Numpy.
np_data = np.array(data)
# Finalmente, convertimos los datos a un DataFrame.
poetas_df = pd.DataFrame({'ID': np_data[:, 1], 'Nombre': np_data[:, 0], 'Revis

poetas_df
```

	ID	Nombre	Revisión
0	4949229	Julia Morilla de Campbell	'''Julia Morilla de Campbell''' ([[Rosario (Ar...
1	5105749	Luis Negreti	{{Ficha de escritor\n|Imagen = NE...
2	4477192	Poldy Bird	{{Ficha de persona\n| padres = Enrique Bird Mo...
3	423422	Ana María Shua	{{Ficha de persona\n|imagen=\n|nombre de nacim...
4	4284479	León Benarós	{{Ficha de persona\n|nombre = León B...
...	...	...	...
634	7887761	Humberto Tejera	{{Ficha de persona\n| nombre = Humberto Tejera...

635	8440353	Mario Molina Cruz	{{Ficha de escritor\n|nombre= Mario Molina Cru...
636	6449649	Luis Ignacio Helguera	{{Ficha de persona\n| nombre = Luis Ignacio He...

Next steps: Generate code with `poetas_df` View recommended plots New interactive sheet

4. Archivos de Excel

Los archivos de excel son a veces nuestros mejores amigos, y otras veces nuestras peores pesadillas. Un archivo en excel (o cualquier otra hoja de caálculo) son formatos muy útiles que permiten compartir información técnica con personas sin preparación técnica, lo que lo vuelve una herramienta muy poderosa para comunicar hallazgos a los usuarios.

Igualmente, la manipulación de datos a través de hojas de cálculo, sin usarlas correctamente 8esto es, programando cualquier modificación) genera normalmente un caos y una fuga de información importante para una posterior toma de desición.

Como buena práctica, si se tiene acceso a la fuente primaria de datos y se puede uno evitar el uso de datos procesados en hoja de calculo, siempre es mejor esa alternativa (como científico de datos o analista de datos). Pero eso muchas veces es imposible.

Vamos a dejar la importación desde `xlsx` a los cursos de *DataCamp* que lo tratan magistralmente. Es importante que, para que se pueda importar desde python o R, muchas veces es necesario instalar librerías extras.