TEMA 4

MÓDULO:
HERRAMIENTAS DE BIG DATA

HERRAMIENTAS DE ANÁLISIS

II. PROGRAMACIÓN EN PYTHON

FERRÁN CARRASCOSA

Licenciado en Matemáticas por la UB
Data Scientist

Institut de Formació Contínua-IL3
UNIVERSITAT DE BARCELONA

ÍNDICE

OBJETIVOS ESPECÍFICOS

- Realizar operaciones de lectura y escritura de datos con Python.
- Saber escoger la estructura de datos de Python adecuada para cada problema.
- Tener las bases para realizar análisis descriptivo mediante tablas y gráficos en Python.
- Desarrollar pequeñas piezas de código en Python.

2. PROGRAMACIÓN EN PYTHON

2.1. INTRODUCCIÓN

Python es una herramienta para la programación de propósito general.

En los últimos años, se ha convertido en uno de los lenguajes de referencia para el Data Science. El motivo es que ha sabido rodearse de un gran ecosistema como SciPy con librerías (numpy, pandas, scipy...) y herramientas (Jupyter, Spyder,...) orientadas al análisis y a la programación matemática.

Los desarrolladores de Python buscan hacer un lenguaje vivo y atractivo para el programador. Prueba de ello es que su nombre es un tributo a la compañía de humor británica Monthy Python.

El núcleo de su filosofía de programación se resume en el "Zen de Python", formado por 20 aforismos escritos por Tim Peters, 19 de los cuales se pueden leer más abajo importando "this". El veinte, dijo Tim Peters que lo diría el creador de Python, Guido van Rossum, pero parece que aún no se ha pronunciado...

```
import this
## The Zen of Python, by Tim Peters
##
## Beautiful is better than ugly.
## Explicit is better than implicit.
## Simple is better than complex.
## Complex is better than complicated.
## Flat is better than nested.
## Sparse is better than dense.
## Readability counts.
## Special cases aren't special enough to break the rules.
## Although practicality beats purity.
## Errors should never pass silently.
## Unless explicitly silenced.
## In the face of ambiguity, refuse the temptation to guess.
## There should be one-- and preferably only one --obvious way to do it.
## Although that way may not be obvious at first unless you're Dutch.
## Now is better than never.
## Although never is often better than *right* now.
## If the implementation is hard to explain, it's a bad idea.
## If the implementation is easy to explain, it may be a good idea.
## Namespaces are one honking great idea -- let's do more of those!
```

Ser *Pythoniano* o *Pythónico* significa utilizar correctamente el código, es decir, programar con un lenguaje simple y fácil de leer.

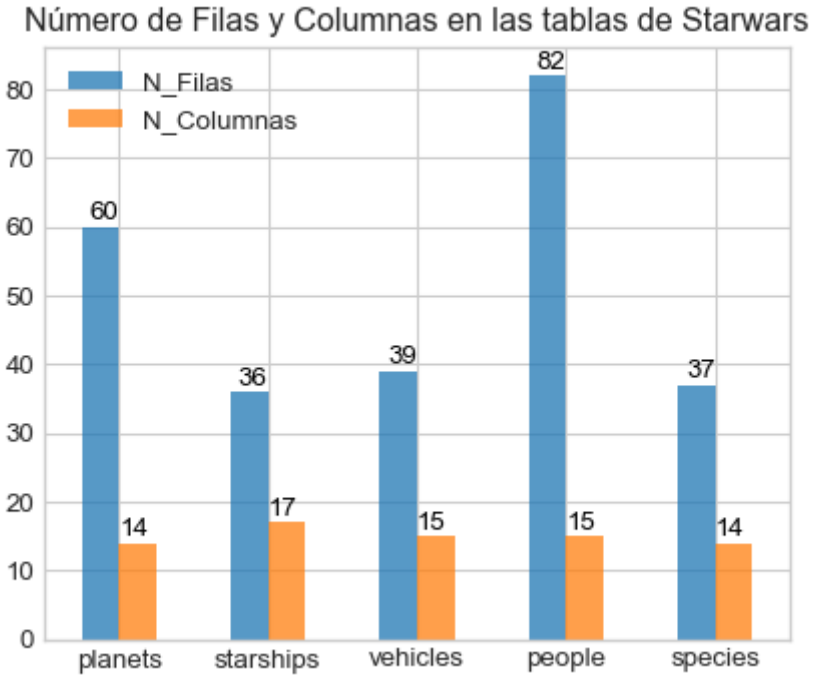
2.2. ACTIVIDAD GUIADA 2

La actividad guiada, que te proponemos, en consonancia con los Pythonianos, es que te diviertas analizando datos relacionados con la saga de Star Wars: Planetas, Naves, Vehículos, Personajes y Especies.

Para ello, contamos con los datos de SWAPI, acrónimo de STAR WARS API, que nos da acceso libre a una colección de datos de la saga.

Estos datos se han descargado y preparado, expresamente, para este curso.

Puedes encontrar el código utilizado en el Anexo: README de Python, capítulo “IMPORTAR DATOS DE STARWARS SWAPI”.



Vemos que los datos están formados por 5 conjuntos. Por ejemplo, “people” contiene 82 personajes descritos mediante 16 variables. Estos personajes tienen columnas numéricas como la altura, el peso, o la edad (en años ABY antes de la Batalla de Yavin). Otras son categóricas, como el género del personaje. Incluso hay columnas en formato de lista, como las películas en las que salió el personaje y los vehículos y naves que condujo.

"Yoda habló de otra."
"La otra de quién habló es tu hermana gemela.»
— Luke Skywalker y Obi-Wan Kenobi

Te acordabas que Luke y Leia eran gemelos? ¿Sabías que su padre (Anakin, posteriormente Darth Vader tenía 22 años cuando los tuvo?

Podrás analizar todo esto y mucho más, en los datos y así convertirte en el auténtico Jedi que sabemos que llevas dentro.

Consulta la Documentación de SWAPI sobre sus tablas y campos.

Y que la fuerza te acompañe...

2.3. ELEMENTOS BÁSICOS DE PYTHON

Aunque se presuponen unos conocimientos iniciales de Python, a continuación, se hace un repaso de Python, Anaconda y notebooks de Jupyter-Colab.

El objetivo es reforzar aquellos elementos más orientados al análisis de datos.

Abre 2.3. Elementos básicos de Python en Colab

2.4. GRÁFICOS CON MATPLOTLIB

La librería gráfica Matplotlib, concebida por John Hunter en 2002, fue construida sobre objetos numpy (arrays N-dimensionales) y, posteriormente, adaptada a objetos Pandas (matrices con vectores datos de distintos tipos).

Actualmente, se apoya en otras librerías, más simples y de aspecto gráfico modernizado con Seaborn y los mismos Pandas.

Los objetos numpy y pandas se expondrán ampliamente en el siguiente apartado, no obstante, dado que el curso presupone ciertos conocimientos de programación, se muestran, ahora, los gráficos para poder utilizarlos en los siguientes apartados como herramienta.

ACTIVIDAD GUIADA 2.2

Consiste en conocer mejor a las especies mediante gráficos.

En concreto, se puede analizar su altura, años de vida, clase de especie (mamífero, reptil, ...) y en cuantas películas ha salido esa especie.

> «¡No puedes llevar a Su Alteza Real allí! Los Hutts son gangsters ...»
> —Quarsh Panaka.

Abre 2.4. Gráficos con Matplotlib en Colab

2.5. COLECCIONES DE OBJETOS

Abre 2.5. Colecciones de objetos en Colab

2.6. NUMPY

El package numpy es la solución más popular dentro de Python para realizar computación científica. Recoge las mejores prácticas introducidas en las "Listas" y organizadas para realizar cálculos de forma eficiente. Se estructuran como vectores o arrays de N, dimensiones de un mismo tipo de dato.

Abre 2.6. Numpy en Colab

2.7. PANDAS

Como abreviación de Panel Data, los objetos pandas dan soporte al análisis de datos con columnas de distinta tipología: categóricas, binarias, numéricas, etc.

Respecto a los numpy, permiten indexar las filas y columnas.

Este tipo de indexación facilita, por ejemplo, realizar análisis de series temporales, no necesariamente con una frecuencia fija.

ACTIVIDAD GUIADA 2.4

Esta vez, se trata de seleccionar el mejor planeta posible para ubicar la academia Jedi. Los parámetros de la búsqueda son:

- Días largos para entrenar mucho.
- Mucha agua para poder refrescarse.
- Poca densidad de población para no ser molestado.
- Buen clima (temperado o tropical).

> «Si existe un auténtico centro del universo, ahora estás en el planeta más alejado de él»
>
> —Luke Skywalker sobre Tatooine el planeta desértico dónde nacieron Anakin y Luke Skywalker.

Abre 2.7. Pandas en Colab

2.8. CONTROL DE FLUJO

Las herramientas de control de flujo permiten automatizar tareas.

En este capítulo, se trabajarán las funciones condicionales y bucles.

DATOS DE EJEMPLO

Este capítulo utilizara ejemplos de vehículos de Star Wars.

Abre 2.8. Control de flujo en Colab

2.9. GESTIÓN DE DATOS

A continuación, presentamos las funciones para la lectura/escritura de datos, cruce y construcción de tablas resumen.

Al final del capítulo, presentamos la forma de manejar datos temporales.

ACTIVIDAD GUIADA 2.5

Se trata de analizar los personajes de la serie:

> «Preferiría ser un monstruo que cree en algo, que sacrificaría todo para mejorar la galaxia, que ser alguien que se quede al margen y mire como si no tuviera repercusión en ellos.»
> —Princesa Leia Organa

Esta actividad consiste en cruzar datos de personajes y planetas para construir descriptivos resumen de los datos de personajes.

Abre 2.9. Gestión de datos en Colab

IDEAS CLAVE

- Python es un lenguaje de propósito general con un conjunto de paquetes que facilitan la computación científica en el ámbito de la ciencia de datos.
- Python permite la visualización de datos.
- Python es un lenguaje perfectamente válido para entornos productivos de tratamientos de datos.
- Python interactúa de forma simple con distintas bases de datos.
- El entorno Jupyter/Colab aportan capacidades interactivas a Python. Permite desarrollar tanto programas, como análisis de datos.
- Se utiliza en todas las fases de análisis de datos:
  - Adquisición y preparación de los datos.
  - Análisis de los datos.
  - Manejo y almacenamiento efectivo de los datos.
  - Comunicación de los resultados: utiliza Jupyter/Colab combinado con Markdown.
  - Aplicación de los resultados obtenidos: utiliza modelos predictivos.

ANEXO: README DE PYTHON

PREPARACIÓN DEL ENTORNO COLAB

Desde Colab, hay que clonar el repositorio cada vez que inicias un nuevo libro.

En los libros se incluye el código necesario para ello.

PREPARACIÓN ENTORNO LOCAL-JUPYTER (OPCIONAL)

CLONAR REPOSITORIO

En local puedes utilizar el mismo proyecto que has clonado en el README DE R.

Para actualizarlo de nuevo, desde consola:

```
cd mbdds_fc20
git pull
cd Python
```

CREAR EN LOCAL UN NUEVO ENVIRONMENT DE ANACONDA

Abrimos una línea de comandos (con *Anaconda 3.0* ya disponible).

- Windows: escribimos Anaconda en el menú Inicio y aparecerá la consola MS-DOS de Anaconda.
- Linux: abrimos "Terminal".

```
conda deactivate
conda create -n mbdds_rpy20 python=3.6.9
conda activate mbdds_rpy20
```

Verifica que se ha creado y está activo.

```
conda info --envs
```

INSTALA LAS LIBRERIAS DE PYTHON

```
cd mbdds_fc20/Python
conda activate mbdds_rpy20
python -m pip install -r requirementsColab.txt
```

PUBLICAR EL KERNEL

Para acceder al nuevo environment desde Jupyter, necesitas publicar el kernel:

```
python -m ipykernel install --user --name mbdds_rpy20 --display-name "mbdds_rpy20"
```

Puede tardar unos minutos en publicarse.

LANZAR ENTORNO JUPYTER NOTEBOOK

Para acceder al servidor Jupyter:

```
conda activate mbdds_rpy20
jupyter notebook
```

Debería abrirse un navegador con acceso a Jupyter desde donde podrás acceder a los notebooks. Habitualmente, el servidor Jupyter se abre en http://localhost:8888/.

Abre Anexo: Readme de Python