Practica 0: Elementos de Python

Bienvenidos a Modelos y Simulación. A lo largo de esta sección práctica exploraremos los conceptos teóricos de clase aplicados en el lenguaje Python utilizando librerías predeterminadas. Algunas aspectos logísticos a cubrir:

- Las clases prácticas serán los Lunes de 16:00 hs. a 17:30 hs.
- Las *Office Hours* donde podrán realizar sus consultas pueden reservarse a través del siguiente link https://calendly.com/tomas-nozica/15min
- Será necesario instalar PyCharm CE para la ejecución de ejercicios, Conda Navigator para el manejo de librerías y Jupyter para explorar los conceptos de clase práctica.

Para esta clase práctica 0, deberán descargar el archivo que les enviaremos el día Lunes 9 de Agosto 17:30 hs. El mismo contiene 3 archivos: `warm-up.py, panda.py` y `matplot.py.`

En todos los casos, encontraran en cada función que deben implementar con el placeholder `pass`. Eliminarlo y escribir allí el código.

warm-up.py

El propósito de estos ejercicios es recordar algunos conceptos sobre el manejo de estructuras de datos, funciones y *control flow* de Python. Se introduce el concepto de Doctests, cada función posee casos que aseguran la calidad de la lógica implementada. En Pycharm, ejecutar estos test es bastante simple. Basta con hacer `click derecho → run doctest` . Para ejecutar normalmente el main, nos posicionamos sobre el y al realizar click derecho deberíamos ver la opción `run`

`list_has_even_size(lst)`

Esta función recibe como parámetro un arreglo de N elementos, retornar `True` si la lista es de tamaño par (múltiplo de 2). Retorna `False` en caso contrario.

`sum_of_elements(lst)`

Esta función recibe como parámetro un arreglo de N números enteros, retornar el valor que se obtiene al sumar cada uno de los elementos.

`remove_elements(array)`

Esta función recibe como parámetro un arreglo de 2 dimensiones de N elementos, retornar un arreglo que solo contenga aquellas filas cuyos elementos no contengan el valor `None`. Recordemos que None no se presenta como un string sino que es una palabra reservada.

`replace_value(array)`

Esta función recibe como parámetro un arreglo de 2 dimensiones de N elementos. Por cada valor de 'x' (como minúscula o mayúscula) remplazarlo por 'o' (respetando mayúsculas y minúsculas). Retornar el arreglo transformado.

panda.py

Panda es una librería muy popular en el mundo de la ciencia de datos. Nos permite explorar, limpiar y procesar información. En Panda, llamamos DataFrame a las tablas que contienen la data. Los siguientes ejercicios tienen como objetivo que apliquen algunas funciones que vimos en clase. Para cualquier consulta dirigirse a la documentación: https://pandas.pydata.org/docs/

En este archivo se provee el `main` de manera tal que al ejecutarlo podamos explorar los resultados. Pueden modificarlo en función de aquello que quieran visualizar.

`set_dataframe(source)`

Esta función recibe como parámetro un diccionario. Empleando la función `DataFrame()` crear el respectivo *dataframe* y retornarlo.

`average_age(data_frame)`

Esta función recibe como parámetro un *dataframe*. Explorando la columna "*Age*" , encontrar el promedio de todos los valores y retornarlo. Se provee un test para validar el resultado.

`people_from(a_country, data_frame)`

Esta función recibe como parámetro un *dataframe*. Empleando la función `loc()` retornar aquellas filas cuyo valor en la columna "Country" sean iguales al string `a_country` pasado por parámetro

matplot.py

Matplot es tambien muy popular en el mundo de la ciencia de datos. Nos permite visualizar la información de un determinado dataset o array. Los siguientes ejercicios tienen como objetivo que apliquen algunas funciones que vimos en clase. Para cualquier consulta dirigirse a la documentación: https://matplotlib.org

NumPy es una librería que contiene funciones matemáticas de "alto nivel" para operar con matrices y vectores. Para acceder a toda la documentación: https://numpy.org

En este archivo se provee el `main` de manera tal que al ejecutarlo podamos explorar los resultados. Pueden modificarlo en función de aquello que quieran visualizar. El gráfico debe visualizarse en una ventana emergente al ejecutar el programa.

`y_values(x, mode)`

Esta función recibe como parámetro un array generado de valores consecutivos dados por la constante X declarada al inicio del programa y '`mode`' donde especificaremos que tipo de función queremos generar. En este caso solo conocemos dos modos:

- '`l`' para una función lineal `y = m*x + b`, los valores de pendiente `m` y ordenada al origen `b` los tomaremos como constantes donde `m = 3` y `b = 2`
- '`q`' para una función cuadrática del tipo `y = x²` .

Retornar el arreglo con los valores `y`. Se proveen dos casos de test para asegurar funcionamiento correcto.

`plot_x_y(x, y, mode)`

Esta función podemos considerarla *dummy* por la implementación que es necesaria para ejecutarla. Pero dado que la decomposición nos asegura calidad y entendimiento, lo planteamos así. Utilizar la función `plot` de la librería `matplot` para generar el gráfico con los valores `x` e `y`