Practica 1

Introducción a Jupyter Notebooks/Google Colab

Los Jupyter Notebooks pueden ser considerados documentos donde se puede escribir tanto texto como código. Los Jupyter Notebooks soportan múltiples lenguajes de programación, incluidos Python, R, Julia y Matlab, los más utilizados en análisis de datos y computación científica.

Los Jupyter Notebooks pueden ser abiertos desde el navegador de la PC (Chrome, Firefox, Edge o cualquier otro).

Se puede instalar un entorno local para trabajar con Jupyter Notebooks mediante la distribución Anaconda de Python. Esta distribución de Python es gratuita, open-source y fácil de usar. Es además multiplataforma, podemos usarla en Mac, Windows o Linux indistintamente. Esta pensada para ser usada por científicos de datos, y por ello, trae incluidos muchos paquetes para trabajar en modelos de aprendizaje automatizado.

Si ya tenemos un entorno local de Python funcionando y solo queremos agregar la posibilidad de utilizar Jupyter Notebooks, sin instalar todo el ecosistema de nuevo, podemos hacerlo directamente a través de la página oficial de Jupyter Notebooks.

Sin embargo lo más cómodo (y lo recomendado para este curso) es utilizar Google Colab. Google Colab es una herramienta muy similar a Jupyter Notebooks, pero no requiere ningún tipo de almacenamiento ya que esta completamente alojado en la nube de Google. Se puede utilizar en conjunto con Google Drive, almacenando los notebooks que escribamos como cualquier otro tipo de documento. Además, esto permite compartir y editar colaborativamente los notebooks. Al igual que la instalación con Anaconda, ya trae incluidos muchos paquetes estándar para el desarrollo de programas orientados al análisis de datos.

Tanto en Jupyter Notebooks como en Google Colab, los documentos están compuestos por una serie de *celdas* (del inglés *cells*). Hay celdas de dos tipos, de texto y de código.

Una celda de texto (como esta que estas leyendo ahora) contiene texto en formato Markdown. Esto permite escribir texto enriquecido con algunos formatos y enlaces de forma muy simple. Una guía para empezar a escribir en este formato puede ser encontrada en (esta página)[https://markdown.es/sintaxis-markdown/]. Adicionalmente, tenemos soporte para LaTeX para escribir ecuaciones matemáticas.

Las celdas de código contiene instrucciones para la computadora, sea el código Python propiamente dicho u otras instrucciones auxiliares. Veamos nuestra primera celda de código:

In [ ]:

```
print("Hola mundo!")
```

```
Hola mundo!
```

Notarás que la celda de código tiene un icono triangular a su izquierda (recordando al ícono "play" de un equipo de música). Al hacer clic en este ícono se correran las instrucciones que haya en la celda (puede ser más de una) y Jupyter Notebooks/Google Colab nos mostrara el resultado debajo. Luego de eso, el ícono cambia a un número que nos irá mostrando el orden en el que corrimos las diferentes celdas. Si acercamos el mouse a este número recuperamos el ícono triangular que nos permitirá volver a correr una celda.

Programación Orientada a Objetos con Python

Realizar un pequeño proyecto para representar mediante clases y objetos juegos con un mazo de carta españolas, como la casita robada, la escoba de 15, el chinchon o el truco argentino.

Conocimiento de base: Un mazo de cartas españolas trae 50 cartas. Estas estan clasificadas segun su *palo*, que

Conocimiento de base: Un mazo de cartas españolas trae 50 cartas. Estas están clasificadas según su *palo*, que puede ser Bastos, Espadas, Copas u Oros. Hay 12 cartas de cada tipo, numeradas correspondientemente. Es común llamar Sota a la carta con el número 10, Caballo a la carta con el número 11, Rey a la carta con el número 12 y As a la carta con el número 1. El mazo de cartas de españolas se completa con dos comodines.

Ejercicio 1: Definir una clase Carta.

Ejercicio 2: Instanciar objetos que representen el as de espadas, un comodín, el 3 de copas y el rey de bastos

Ejercicio 3: Definir un método que nos permita imprimir las cartas como lo haríamos naturalmente.

Ejercicio 4: Escribir un método que nos permita comparar cartas por igualdad

Ejercicio 5: Escribir una clase mazo, que construya el mazo de cartas españolas. Escribir un método que devuelve cuantas cartas hay en el mazo.

Ejercicio 6: Escribir un método en la clase Mazo que *mezcle* el mazo. Puede ser de utilidad el módulo `random` de la bilbioteca estándar de Python.

Ejercicio 7: Implementar en la clase Mazo, un método que permita sacar una carta específica del mazo, y que devuelva `True` si la carta estaba presente o `False` si no lo estaba.

Ejercicio 8: Implementar un método para robar una carta del mazo, es decir, para sacar aquella que se encuentra primera.

Ejercicio 9: Implementar un método que nos permita saber si quedan cartas en el mazo

Ejercicio 10 Escribir una clase Mano, que represente la mano de un jugador en algun juego de cartas. Tener en cuenta que necesitaremos los métodos `sacar_carta` y otros ya definidos. Además, necesitaremos asociar el nombre del jugador que tiene esta mano

Ejercicio 11: Necesitaremos que una mano tenga funcionalidad para agregar cartas a la mano y sacar cartas de la mano. ¿Cuantos métodos debemos definir?

Ejercicio 12: Agregar al mazo un método para repartir cartas. El metodo deberia recibir una lista de manos y la cantidad de cartas a repartir en cada mano.

Ejercicio 13: Agregar funcionalidad para imprimir una mano, mostrando a quien pertenece y que cartas contiene

Ejercicio 14: Agregar una clase Juego que represente un juego con cartas españolas.

Ejercicio 15: Heredar una clase TrucoArgentino que represente un juego de truco argenino para dos jugadores. Recordar:

- **Antes de empezar a jugar se deben quitar los 8 y los 9 de todos los palos, y los dos comodines.**
- **se deben inicializar dos manos de tres cartas cada una. Se reciben por parametros al constructor los nombres de ambos jugadores.**

Ejercicio 16: Implementar un método `.gana_envido()` que devuelva el nombre del jugador que tiene mas punto de envido.

- **Asumimos que nuestros jugadores son muy malos en el truco y nunca mienten.**
- **Jugamos sin flor.**
- **Recordar que para calcular el envido, si un jugador posee dos o más cartas de igual palo, los puntos de envido equivale a la suma del puntaje de dos cartas del mismo palo elegidas por el jugador más veinte puntos (10, 11 y 12 no suman). Jugamos asumiendo que mano1 es mano del partido (es decir, gana el envido en caso de empate).**