"El saber de mis hijos
hará mi grandeza"

Universidad De Sonora

División de Ciencias Exactas y Naturales

Licenciatura En Física

Física Computacional I

Reporte de Actividad 2

"Jupyter Notebook como entorno de programación"

César Omar Ramírez Álvarez

Profr. Carlos Lizárraga Celaya

Hermosillo, Sonora Febrero 7 de 2018

1 Introducción

Jupyter Notebook es una aplicación web que permite crear y compartir documentos que contienen código fuente, ecuaciones, visualizaciones y texto explicativo. Entre sus usos está la limpieza y transformación de datos, la simulación numérica, el modelado estadístico, el aprendizaje automático y mucho más.

Jupyter permite interactuar con varios lenguajes de programación, en este, usaremos Python, un lenguaje de programación bastante simple y poderoso, con acceso a una gran variedad de librerias para procesamiento de datos. Entre estas, está Pandas, una biblioteca que nos da acceso a estructuras de datos muy poderosas para manipular datos.

Pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para que trabajar con datos "relacionales" o "etiquetados" sea fácil e intuitivo. Su objetivo es ser el componente fundamental de alto nivel para hacer un análisis práctico y real de datos en Python.

En esta ocasión y como producto de una serie de actividades realizadas con el uso de Jupyter Notebook con el lenguaje Python, aprendimos usos básicos de éste lenguaje que mas adelante se ejemplificarán.

2 Jupyter Notebook como entorno de programación

El Jupyter Notebook es una aplicación web de código abierto, desarrollada utilizando lenguaje HTML, generalmente esta herramienta es utilizada para el aprendizaje del lenguaje de programación Python.

2.1 Características

Entre las muchas características de Jupyter Notebook podemos destacar:

- De fácil instalación gracias a estar presente en la Suite Anaconda Distribution.
- Posee una avanzada interfaz web que permite combinar código fuente, textos, formulas, figuras y multimedia en un solo documento.
- La integración de diverso tipos de información nos permite dar explicaciones más adecuada de nuestros programas o de los conceptos que estemos aprendiendo.

- Permite el acceso desde cualquier lugar sin necesidad de instalación de otros servicios, ya que funciona como cliente servidor. De igual manera, Se puede ejecutar en un escritorio local o en servidor remoto.
- Aunque el lenguaje de programación fundamental en Jupyter Notebook es Python, esta aplicación también es compatible con más de 40 lenguajes, entre los que destacan R, Julia y Scala.
- Permite el intercambio de documentos de Jupyter a través de servicios de terceros.
- Podemos ejecutar y visualizar imágenes, videos, LaTeX y JavaScript, además de manipular los resultados de los mismos en tiempo real.
- Cuenta con un administrador de documentos avanzado, que permite visualizar los archivos compatible con Jupyter Notebook que esten alojados en nuestro equipo.
- Los documentos realizados en Jupyter Notebook se pueden exportar a diferentes formatos estáticos incluyendo HTML, reStructeredText, LaTeX, PDF y presentaciones de diapositivas.
- Es compatible con nbviewer el cual permite que portar nuestros documentos de Jupyter Notebook a la nube como una página web estática, la cuál podrá ser visualizada por cualquiera sin necesidad de instalar el Jupyter Notebook .

3 Resumen General

Para poder hacer uso de Jupyter Notebook, primeramente se seleccionó la carpeta de *"Física Computacional"* posteriormente de esa carpeta se abre una terminal y escribimos *Jupyter Notebook* inmediatamente se abre una pestaña de nuestro navegador, seleccionamos la opción Phyton3.0 y nombramos la pestaña como *Actividad 2*, teniendo esto ya se esta listo para programar.

En esta ocasión, se trabajó con datos proporcionados por el Servicio Meteorológico Nacional, se visitó su sitio web y se tomaron datos meteorológicos de cada 60 *min* del municipio de Chínipas de Almada, Chihuahua. Los datos fueron guardados en la carpeta de Actividad 2. Todo esto para poder que aparezcan en nuestro Jupyter Notebook.

3.1 Bibliotecas

Antes de iniciar con cualquier tipo de código es de suma importancia y recomendable agregar las bibliotecas para que se carguen a la memoria de las celdas. Se agregaron 3 muy importantes: Pandas, Matplotlib.pyplot y Numpy. Siendo Pandas usada para la manipulación en el manejo de datos y análisi para la programación en Phyton, Numpy un paquete indispensable que permite utilizar matrices, transformadas e inclusive códigos externos de otros lenguajes de programación y por último Matplotlib.pyplot utilizada para la realización de gráficas en 2 dimensiones para Phyton.

La manera de cargarlas a las celdas es la siguiente, es recomendable abreviarlas.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Para correr una celda con códigoes necesario aplicar la combinacion de las teclas *Shift+Enter*, esta opción crea inmediatamente una celda abajo para seguir programando.

3.2 Comandos

3.2.1 Lectura de Documento

Digamos que queremos leer el documento de *Chínipas de Almada, Chihuahua*, el código para hacerlo seria de la suiguiente manera:

```
df0 = pd.read_csv('chinipas.txt', skiprows=4, sep='\s+')
```

Esto es posible mediante Pandas, debemos nombrar con una variable el documento que guarda la información, se sigue de un igual y la refrencia de lectura con Pandas, el nombre del documento y la información o encabezados que nos debemos de saltar del documento (por lo regular antes de los datos vienen encabezados).

Para poder imprimir datos leídos del documento es necesario utilizar la función *head()*, que si es usada así tal cual imprime las primeras 5 lineas pero si colocamos en el paréntesis un número entero n, esas n lineas imprimirá.

Como ejemplo las primeras 5 lineas del documento:

	DD/MM/AAAA	HH:MM	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
0	25/01/2018	23:00	153	110	9.69	27.2	28.0	3	963.6	0.0	302.3
1	26/01/2018	00:00	244	191	3.94	12.8	24.3	9	964.2	0.0	25.7
2	26/01/2018	01:00	186	259	0.27	4.2	19.9	18	964.7	0.0	0.0
3	26/01/2018	02:00	103	117	0.48	4.3	15.6	24	965.2	0.0	0.0
4	26/01/2018	03:00	294	275	0.21	4.4	12.5	32	965.9	0.0	0.0

3.2.2 Estructura y Tipo de Datos

Con Pandas podemos hacer que se acomoden los textos a leer por Phyton, ya que no se saben que estructura guardan. Por lo que se utiliza una función y se crea una varibale que guarde los datos y la estructura. La función quedaría de la siguiente manera:

```
df = pd.DataFrame(df0)
```

Para saber que tipo de datos esta leyendo pandas, podemos hacer uso de la siguiente función y nos despliega la tabla de acontinuación:

```
df.dtypes

DD/MM/AAAA     object
HH:MM          object
DIRS            int64
DIRR            int64
VELS          float64
VELR          float64
TEMP          float64
HR              int64
PB            float64
PREC          float64
RADSOL        float64
dtype: object
```

3.2.3 Mezclar y Eliminar Columnas

Para mezclar la información de columnas y crear una nueva con los datos, se utilizó la siguiente función con nuestro ejemplo:

```
df['FECHA'] = pd.to_datetime(df.apply(lambda x: x['DD/MM/AAAA'] + ' ' + x['HH:MM'], 1)
```

Donde FECHA será la nueva columna en la que se mezclaran fecha y hora.
Para el caso de eliminación se ocupa la siguiente función:

```
df = df.drop(['DD/MM/AAAA', 'HH:MM'], 1)
```

Donde eliminaremos la fecha y lo hora.

3.2.4 Análisis Exploratorio de Datos

El análisis exploratorio de datos, ofrece una tabla en la que se muestran una descripción cantitativa con la mucha o poca cantidad de datos del documento,es decir, nos muestra la media, estándar, los cuartiles, la mínima y máxima de ada columna. Se obtien con la función siguiente, se muestra el ejemplo para el documento que estamos tratando:

```
df.describe()
```

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
count	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.0	166.000000
mean	202.475904	191.614458	2.149578	7.766867	17.034337	34.289157	965.470482	0.0	187.823494
std	74.447680	79.155134	2.501353	5.354334	9.087679	17.740233	2.453106	0.0	273.857529
min	0.000000	27.000000	0.000000	0.000000	0.600000	3.000000	960.500000	0.0	0.000000
25%	154.000000	114.500000	0.512500	4.300000	9.425000	16.000000	963.800000	0.0	0.000000
50%	212.000000	203.000000	1.140000	6.100000	15.900000	36.000000	965.450000	0.0	0.000000
75%	249.500000	247.750000	2.865000	8.975000	25.200000	51.000000	967.400000	0.0	394.500000
max	353.000000	350.000000	14.350000	30.600000	33.500000	65.000000	970.800000	0.0	792.500000

3.2.5 Datos Condicionados en Tabla

Para mostar ciertos datos en una tabla, es decir, condicionar datos. se hace uso de lo siguiente, haciendo seguimiento con nuestro documento:

```
df_tmp = df[df.TEMP > 24]
df_select = df_tmp[df_tmp.TEMP < 25]
df_select
```

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL	FECHA
1	244	191	3.94	12.8	24.3	9	964.2	0.0	25.7	2018-01-26 00:00:00
25	219	232	3.72	12.3	24.8	13	963.3	0.0	28.5	2018-01-27 00:00:00
100	177	256	2.14	11.9	24.2	18	966.4	0.0	0.0	2018-01-30 03:00:00
139	219	221	3.41	7.1	24.5	22	965.1	0.0	672.0	2018-01-31 18:00:00
148	141	81	4.95	22.1	24.3	26	963.7	0.0	0.0	2018-02-01 03:00:00

Como se puede notar estamos presentando una tabla con valores de temperatura entre cierto rango.

3.2.6 Valores Promedio

Si queremos conocer el valor promedio de los datos, recurrimos a la función que acontinuación se presentará y nos dará el promedio por cada columna. En el ejemplo quedaron de la siguiente manera:

```
df.mean()

DIRS      202.475904
DIRR      191.614458
VELS        2.149578
VELR        7.766867
TEMP       17.034337
HR         34.289157
PB        965.470482
PREC        0.000000
RADSOL    187.823494
dtype: float64
```

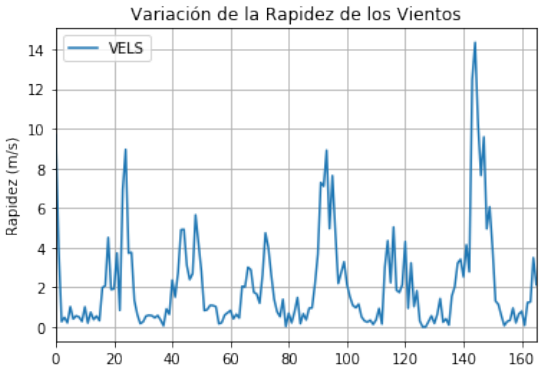
```
# Calcula el promedio de las Temperaturas
df.TEMP.mean()

17.03433734939759
```

3.3 Graficación

Con el paquete de matplotlib podemos crear diversas graficas. Donde ejemplo de un código de ellas es el siguiente:

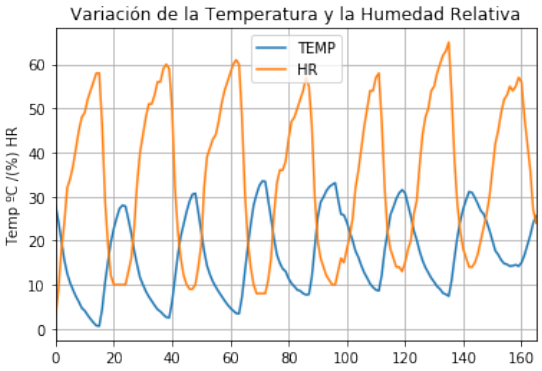
```
# Gráfica de la rapidez de los vientos (m/s)
plt.figure(); df.VELS.plot(); plt.legend(loc='best')
plt.title("Variación de la Rapidez de los Vientos")
plt.ylabel("Rapidez (m/s)")
plt.grid(True)
plt.show()
```

Donde figure indica la realización de una gráfica, df.VELS.plot es lo que se va a graficar, plt.legend hrá que las leyendas se acomoden donde mejor parezcan, title, ylabel y xlabes son el titulo de la gráfica y sus ejes, grid muestra la cuadrícula y plt.show() imprime. La gráfica del código anterior es la siguiente:



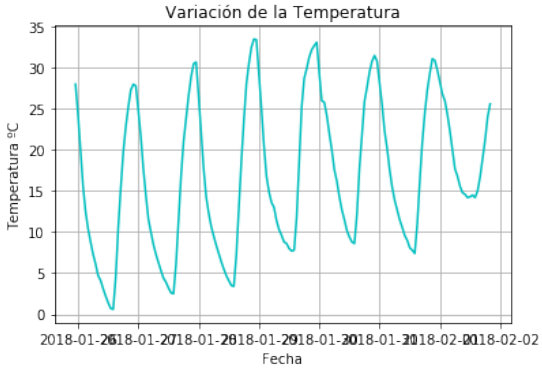
Ahora bien, para tener dos valores en una misma gráfica tenemos como ejemplo lo siguiente:

```
# Gráfica de Temperatura y Humedad Relativa
df1 = df[['TEMP','HR']]
plt.figure(); df1.plot(); plt.legend(loc='best')
plt.title("Variación de la Temperatura y la Humedad Relativa")
plt.ylabel("Temp ºC /(%) HR")
plt.grid(True)
plt.show()
```



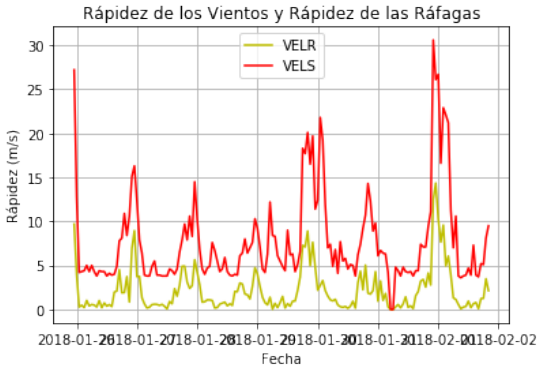
Aquí podemos notar que mientras más humedad exista, menor sera la temperatura.

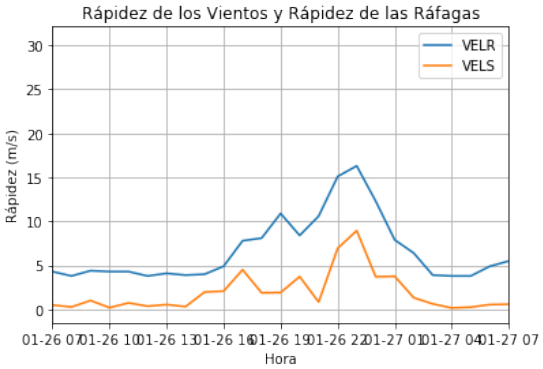
Seguiendo con los ejemplos de nuestro Documento, se nos pidió que hicieramos las siguientes gráficas:



Que nos muestra como cambia la temperatura a través del tiempo. (De los días en los que se tomó la muestra de datos).

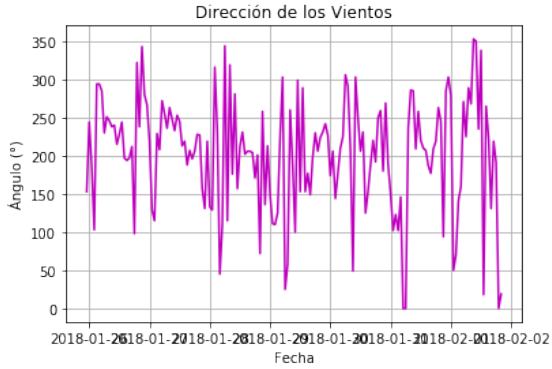
A continuación se muestran las gráficas de la rápidez de los vientos y de las ráfagas, tenemos a la izquierda la general (de todos los datos del documento). Y a la derecha una particular (de datos tomados en un solo día), cabe resaltar que las horas en ésta estan transformadas, ya que los datos son tomados desde hora central, por lo que las 07:00:00 hr de nuestro horario, son las 00:00:00 hr de la hora central.





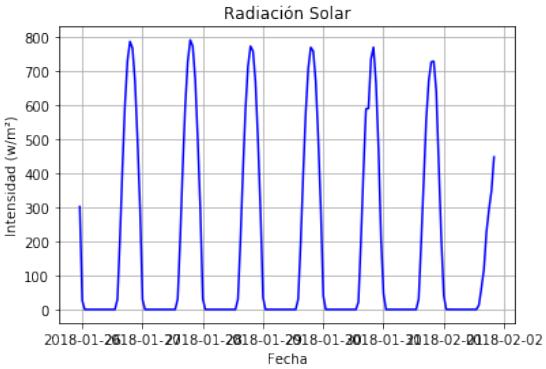
Podemos rescatar de la gráfica de la derecha que las horas del día en las que tenemos más vientos es desde las 12:00:00 hrs hasta las 18:00:00 hrs medidos desde nuestro sistema de referencia.

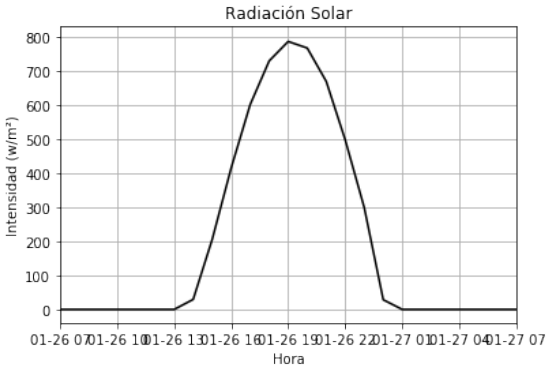
La siguiente gráfica muestra la dirección de los vientos a lo largo de que transcurre el tiempo:



Notando que los vientos dominantes andan en un promedio de 200 °

En cuanto a la radiación solar tenemos las siguientes gráficas, a la izquierda la general y a la derecha la de un día en específico:





Agregando que según nuestro sistema de referencia las horas del día con mas radiación solar son a mediodía, ya que el Sol se encuentra paralelo a los detectores y es por ello que marca mas radiación.

También se nos pidió que calcularamos la diferencia de temperaturas entre la máxima y la mínima de un día, y el resultado con su respecto código es el siguiente:

```
df2 = df.loc[2:24,['TEMP','FECHA']]

tmin = df2.min()

tmax = df2.max()

tmax - tmin

TEMP                  27.4
FECHA    0 days 22:00:00
dtype: object
```

Por último se realizó un análisis exploratorio de datos y el resultado fue el siguiente:

```
df.describe()
```

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
count	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.0	166.000000
mean	202.475904	191.614458	2.149578	7.766867	17.034337	34.289157	965.470482	0.0	187.823494
std	74.447680	79.155134	2.501353	5.354334	9.087679	17.740233	2.453106	0.0	273.857529
min	0.000000	27.000000	0.000000	0.000000	0.600000	3.000000	960.500000	0.0	0.000000
25%	154.000000	114.500000	0.512500	4.300000	9.425000	16.000000	963.800000	0.0	0.000000
50%	212.000000	203.000000	1.140000	6.100000	15.900000	36.000000	965.450000	0.0	0.000000
75%	249.500000	247.750000	2.865000	8.975000	25.200000	51.000000	967.400000	0.0	394.500000
max	353.000000	350.000000	14.350000	30.600000	33.500000	65.000000	970.800000	0.0	792.500000

Con esto se finalizó la actividad, a lo que podemos llamar la primera interacción con el lenguaje de programación Python, el uso de Jupyter Notebook y todas las bibliotecas.

4 Conclusión

A manera de conclusión de esta práctica puedo decir que fue bastante conocimiento básico el que se adquirió con el desarrollo de las actividades planteadas, procurando mas retroalimentación en las siguientes.

El hecho de trabajar con datos proporcionados con el Servicio Meteorlógico Nacional, nos hace reflexionar de la multidisciplinariedad de la Física.

5 Apéndice

1. ¿Cuál es tu primera impresión de Jupyter Notebook?

   En un principio pensé que jamas iba a poder llegar a saber, pues lo veía muy complicado y más aún por que mi inglés no es muy bueno, pero con el desarrolo de actividades fui comprendiendo y me parece muy padre.

2. ¿Se te dificultó leer código en Python?

   Sí, la verdad me fue muy difícil comprender ciertas partes de código, pero con lecturas e investigación logré poder modificar e incluso generar nuevo código para la actividad.

3. ¿En base a tu experiencia de programación en Fortran, que te parece el entorno de trabajar en Python?

   Se ve muy bien, pareciera que Python es de mejor manejo (por ahora), pero si difiere mucho con el código a usar con Fortran.

4. A diferencia de Fortran, ahora se producen las gráficas utilizando la biblioteca Matplotlib. ¿Cómo fue tu experiencia?.

   Me gustó mucho poder ver las gráficas y la manera rápida y sencilla de crearlas y modificarlas. Con Fortran no me quedaron muy en claro el uso para la graficación.

5. En general, ¿qué te pereció el entorno de trabajo en Python?

   Muy padre, espero con el desarrollo de las demás actividades poder lograr "dominar" este lenguaje. Espero mucho de él.

6. ¿Qué opinas de la actividad? ¿Estuvo compleja? ¿Mucho material nuevo? ¿Que le faltó o que le sobró? ¿Qué modificarías para mejorar?

   La actividad en sí me gusto mucho, trabajar con datos me gusta, ya que de alguna u otra manera logras inferir buenas conclusiones de ellos. No estuvo dificil, quiza en un principio si, pero despues se fueron complementando y entendiendo las cosas. Si fue material nuevo pero se logró entender, creo que fue muy completa.

7. ¿Comentarios adicionales que desees compartir?

   La verdad estoy muy contento con aprender este nuevo lenguaje.

6 Bibliografía

- Jupyter notebook: documenta y ejecuta código desde el navegador | Desde Linux. (2018). Obtenido de: https://blog.desdelinux.net/jupyter-notebook/
- Jupyter: Data Science aplicada - Paradigma. (2018). obtenido de: https://www.paradigmadigital.com/dev/jupyter-data-science-aplicada/
- Guía de inicio rápido de Jupyter Notebook. (2018). Obtenido de: https://live.osgeo.org/es/quickstart/jupyter_quickstart.html