Universidad de Sonora

Licenciatura en Física

Física Computacional I

Actividad 2 - Python y Jupyter

Alumno:
José Gabriel Navarro I.

Profesor:
Carlos Lizarraga Celaya

07 de Febrero de 2017

En el presente reporte se habla acerca de la segunda actividad realizada para la clase de Física Computacional I, el cual abarca los conceptos de Python y Jupyter, y las distintas funciones, ventajas, desventajas y bondades de estos.

1 Introducción e Investigación

Primeramente y como introducción, ¿qué es Python y que es Jupyter?. Python es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, es decir, el código se escribe en un lenguaje legible por humanos y se traduce renglón por renglón, no lo compila, lo lee mientras se escribe. Posee una licencia de código abierto, denominada Python Software Foundation License y es compatible con la licencia general GNU. Una de las librerias mas usadas para el análisis de datos en Python es la llamada **panda**. Panda es un paquete de Python que provee estructuras para datos de una manera rapida y flexible, diseñado con el proposito de que sea mas fácil e intuitivo la organización de datos. Panda permite analizar datos de distintas formas, por columnas de tipo SQL o Excel, ordenados o desordenados, etcétera. Las principales formas de lectura de datos son ***series*** (para datos unidimensionales) y ***dataframe*** (para datos bidimencionales).

Ahora bien, Jupyter es una aplicación web de código abierto que permite la creación y compartir documentos que contienen código, ecuaciones, visualizaciones de imagenes y texto. Soporta hasta 40 tipos distintos de lenguajes, entre ellos R y Python. En esta clase se utilizara para poder realizar análisis de datos en Python, como se mostrará mas adelante en el reporte. En el caso de esta actividad, se utilizo para el análisis de eventos climatologicos en un cierto estado del país.

2 Actividades a realizar

2.1 Comandos principales de pandas y Python

Antes de realizar el análisis de datos correspondiente al estado elegido, se deben importar las bibliotecas correspondientes para poder hacerlo. Estas bibliotecas son la de "panda" ya mencionada anteriormente y la de matplotlib.pyplot, que se usa para graficar.

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

Una vez importadas las librerías a utilizar, primeramente vamos a leer los datos del archivo del estado elegido, en este caso: Rio Tomatlan. Esto lo hacemos mediante el comando *read_csv*, indicando el nombre del archivo, los renglones que se deben saltar, y el tipo de separación que existe entre los datos, como en el archivo están separados por mas de un espacio, se usa el termino: '\s+'.

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

Para revisar si el archivo fue leído con éxito, podemos utilizar el comando *df0.head()*, que mostrara los primeros cinco renglones de los datos leídos, al colocar un numero entre los paréntesis, ese es el numero de renglones que se mostraran en pantalla.

	DD/MM/AAAA	HH:MM	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
0	25/01/2018	22:00	184	211	2.13	18.4	26.0	68	995.8	0.3	224.0
1	25/01/2018	23:00	0	236	0.00	0.0	26.2	71	995.9	0.0	126.2
2	26/01/2018	00:00	0	305	0.00	0.0	26.0	71	996.0	0.0	37.8
3	26/01/2018	01:00	0	343	0.00	0.0	25.2	74	996.4	0.0	0.0
4	26/01/2018	02:00	0	320	0.00	0.0	24.1	77	996.9	0.0	0.0

```
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
```

Ahora vamos a darle forma de datos, es decir, panda le dará una clasificación a cada tipo de dato que hay en el archivo. Esto se hace por medio del comando *df = pd.DataFrame(df0)*, si compila bien, al introducir el código *df.dtypes*, aparecerán en pantalla el tipo de datos que existe en el archivo. Como se puede observar, los que son números los toma como int64, y los que tienen punto decimal los toma como float64. Ademas de esto también aparece la clasificación de object, que es el tipo que se le pone a los datos que panda no reconoce. (Sin embargo si existe un tipo para la fecha).

Para poder realizar gráficas de los datos contra el tiempo, combinaremos dos columnas, siendo estas la de fecha y hora, esto con el comando de df, seguido por el nombre de la nueva columna, en donde indicamos que tomaremos las columnas de Fecha y de Hora. Al combinarlas indicamos que el día debe de aparecer primero en el formato, ya que en el archivo viene indicado así. Ya combinadas, eliminamos las columnas de fecha y hora, esto mediante el código: *df.drop*, indicando las columnas a eliminar.

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL	FECHA
0	184	211	2.13	18.4	26.0	68	995.8	0.3	224.0	2018-01-25 22:00:00
1	0	236	0.00	0.0	26.2	71	995.9	0.0	126.2	2018-01-25 23:00:00
2	0	305	0.00	0.0	26.0	71	996.0	0.0	37.8	2018-01-26 00:00:00
3	0	343	0.00	0.0	25.2	74	996.4	0.0	0.0	2018-01-26 01:00:00
4	0	320	0.00	0.0	24.1	77	996.9	0.0	0.0	2018-01-26 02:00:00
5	44	53	3.34	18.9	23.8	78	997.5	0.0	0.0	2018-01-26 03:00:00
6	36	45	5.50	17.7	23.1	80	998.0	0.0	0.0	2018-01-26 04:00:00
7	14	30	5.63	16.4	22.9	81	997.9	0.0	0.0	2018-01-26 05:00:00
8	359	356	4.30	16.5	22.6	81	997.7	0.0	0.0	2018-01-26 06:00:00
9	325	25	1.02	15.3	22.5	81	997.3	0.0	0.0	2018-01-26 07:00:00

Además de esto, la librería de panda nos ofrece otro tipo de análisis de datos, entre ellos varios datos estadísticos, como lo es el numero de datos, la media, la desviación estándar, mínimo y máximo y los cuartiles. Se puede dar un análisis general de todos los datos con el comando *df.describe()*. Sin embargo, si se desea un análisis especifico de todos los datos, o un análisis especifico de un dato en especifico, podemos utilizar *df.DATO.VALOR*, donde dato es el dato en especifico y valor el valor en especifico, o también puede pedirse entre un rango de valores utilizando *df_ select*, por ejemplo:

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
count	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000
mean	64.963855	131.777108	2.132952	9.692169	24.340361	67.132530	995.138554	0.009639	152.165663
std	93.298731	110.385022	3.273923	9.384778	3.217829	13.642866	1.827423	0.076459	240.971915
min	0.000000	1.000000	0.000000	0.000000	19.400000	33.000000	991.200000	0.000000	0.000000
25%	0.000000	49.000000	0.000000	0.000000	21.900000	58.000000	993.800000	0.000000	0.000000
50%	33.500000	71.500000	0.075000	11.250000	23.850000	70.000000	995.100000	0.000000	0.100000
75%	67.500000	186.750000	3.505000	16.500000	26.075000	78.750000	996.400000	0.000000	214.475000
max	359.000000	356.000000	17.470000	35.700000	32.900000	87.000000	999.200000	0.800000	835.000000

(a) Un análisis general de los datos

```
# Calcula el promedio de las columnas, excepto en la FECHA (que no tendría sentido)
df.mean()
```

```
DIRS       64.963855
DIRR      131.777108
VELS        2.132952
VELR        9.692169
TEMP       24.340361
HR         67.132530
PB        995.138554
PREC        0.009639
RADSOL    152.165663
dtype: float64
```

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

```
24.340361445783135
```

(b) Análisis de una medida en especifica.

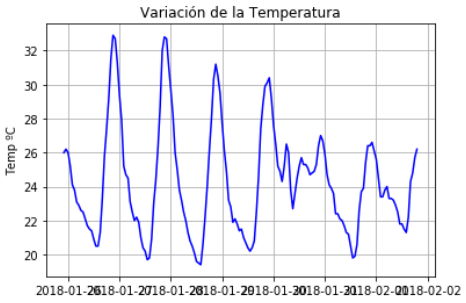
El paquete de matplot, permite la realización de gráficas, y para utilizarlo solamente se debe usar el comando *plt.figure*, en donde se indica que es lo que se va a graficar y

contra quien, en donde se puede especificar también el nombre del gráfico, así como el nombre de sus ejes, plt.grid indica que si se desea usar la cuadricula de fondo y plt.show muestra la gráfica en pantalla:

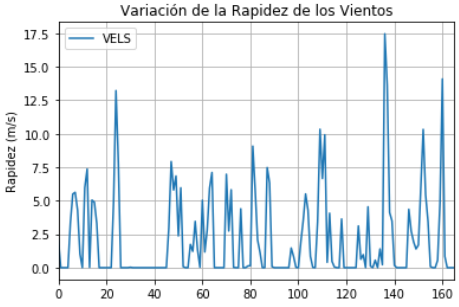
```
plt.figure(); df.RAPVIENTO2.plot(); plt.legend(loc='best')
plt.title("Variación de la Rapidez de los Vientos")
plt.ylabel("Rapidez (m/s)")
plt.grid(True)
plt.show()
```

También se puede escoger en especifico que dato va en cada eje, para de esta manera realizar cualquier gráfica que se desee. Además, se puede cambiar el formato de la gráfica, como lo es su color. Esto lo podemos hacer en la linea de código: *plt.plot_ date(x=df.DATO, y=dfDATO., fmt="COLOR-")*, en donde DATO es la columna de DATOS a utilizar y COLOR la primera letra del color que queremos.

Con estas herramientas, podemos realizar análisis de datos muy extensos, facilitando mas así nuestro trabajo. A continuación se responden las preguntas correspondientes a la practica.



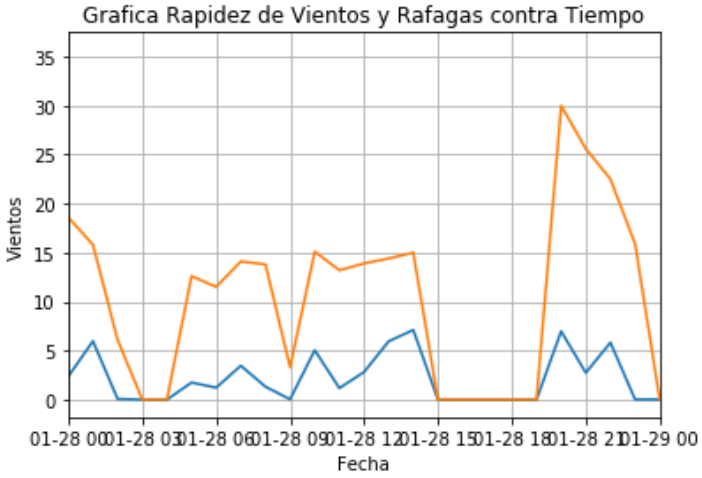
(a) Gráfica de la Variación de la Temperatura



(b) Gráfica de la variación de la rapidez de los vientos

2.2 Analisis de Datos de Río Tomatlan

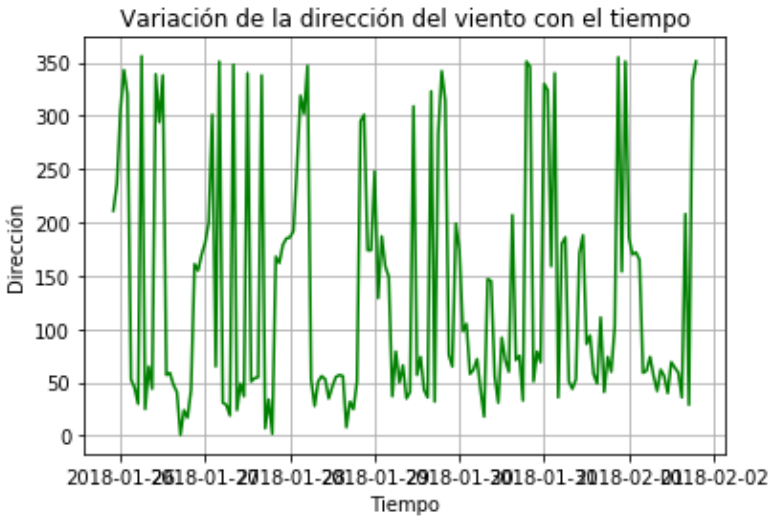
1. Crear una gráfica que muestre la rapidez de los vientos y la rapidez de las ráfagas, como funciones del tiempo. ¿Cuáles son las horas del día con más viento?



Para el análisis de velocidad de ráfagas y vientos, se eligió el rango entre los días 28 de enero de 2018 y 29 de enero de 2018. Como se puede observar en la gráfica existen varios registros de ciertos picos en la velocidad del viento y ráfagas, siendo en el rango de 20:00 a 00:00 horas en donde, tanto como la velocidad de los vientos, como de las ráfagas era el mayor en ese día. Además de esto, se puede observar como en las tardes no hay mucho viento.

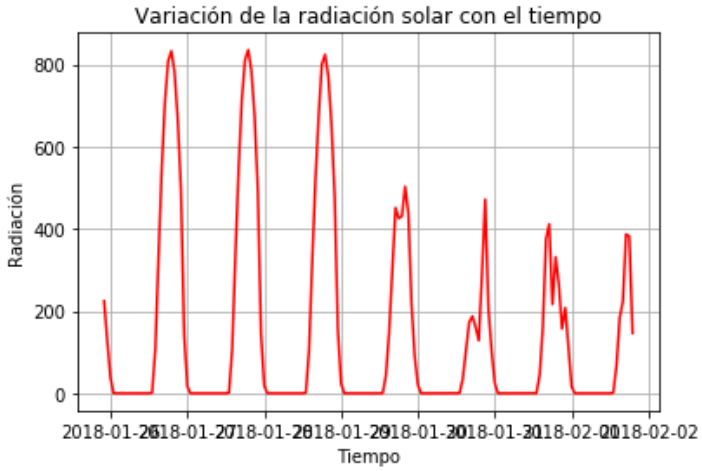
2. Crear una gráfica con la dirección de los vientos como función del tiempo y comentar sobre los vientos dominantes en el sitio de estudio.

La siguiente gráfica muestra el comportamiento de la dirección de los vientos durante 5 días, en donde como se puede observar, la mayoría de las veces o llegan hasta casi los 360°, pero en otras horas del día llegan a casi 0°. Este patrón se puede observar sobre todo entre las ultimas horas del día y las primeras horas del siguiente día, en donde los vientos son dirigidos hacia el sur, mientras que en las tardes, los vientos están dirigidos hacia el norte/noroeste.



3. Muestre el comportamiento de la Radiación Solar como función del tiempo. ¿Qué puedes comentar?

Se puede observar varios registros con 0, esto se debe a que en esas horas aun es de noche, por tanto no hay Sol, y pues no hay radiación solar. Otra cosa a notar es que existen días con mucha radiación solar, siendo los primeros días lo que llegan hasta casi 800, mientras que los últimos días llegan hasta 300.



4. ¿Cuál es el lapso de temperatura diaria? (Diferencia entre la temperatura máxima y la mínima).

Se tomara el mismo rango de días que se tomo para el análisis de vientos y ráfagas, es decir del 28 de enero de 2018 y al 29 de enero de 2018.

Para esta evaluación, se creo otro dataframe con solo las temperaturas dentro del rango indicado, como se muestra en la figura.

```
# Selecciona los renglones con Fecha entre '2018-01-28','2018-01-29' >
df_fch = df[df.FECHA >= '2018-01-28 00:00:00']
df_select = df_fch[df_fch.FECHA < '2018-01-29 00:00:00']
df_select
```

	DD/MM/AAAA	HH:MM	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL	FECHA
50	28/01/2018	00:00	197	186	2.37	18.6	29.7	36	991.5	0.0	18.3	2018-01-28 00:00:00
51	28/01/2018	01:00	180	192	5.97	15.8	28.2	38	992.1	0.0	0.0	2018-01-28 01:00:00

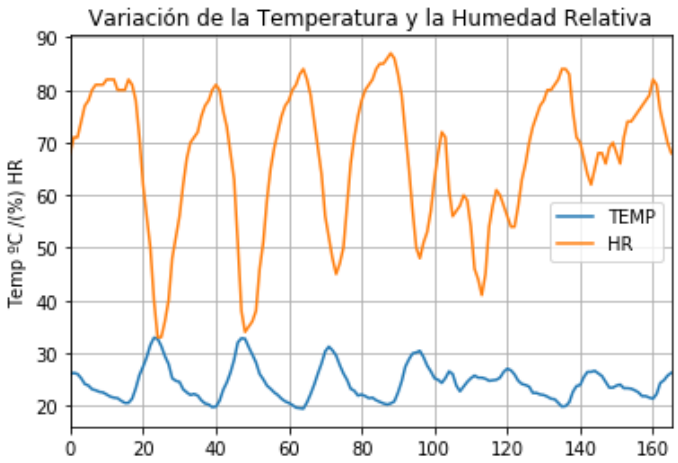
Entonces, ahora con el nuevo dataframe, calculamos el maximo, el minimo, y los restamos, obteniendo que el lapso es de 11.8.

```
In [19]: df_select.TEMP.max()-
Out[19]: 31.199999999999999

In [20]: df_select.TEMP.min()
Out[20]: 19.399999999999999

In [21]: df_select.TEMP.max()-df_select.TEMP.min()
Out[21]: 11.800000000000001
```

5. ¿Puedes comentar sobre la relación entre la temperatura y la humedad relativa?

Como se puede observar por la grafica, la temperatura cambia entre un rango de 32°-18°, sin embargo, el cambio de la humedad es muy grande en los intervalos de tiempo, como se puede observar varia mucho, desde 80° hasta casi 32°.

Cuando la temperatura es muy baja, la humedad es alta, y cuando la temperatura es alta, la humedad es baja.



6. Realiza el análisis exploratorio de datos, que resuma el sitio estudiado (Usar la función describe() sobre tu data frame.

A continuación se muestra el análisis exploratorio de los datos de Río Tomatlan:

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RADSOL
count	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000	166.000000
mean	64.963855	131.777108	2.132952	9.692169	24.340361	67.132530	995.138554	0.009639	152.165663
std	93.298731	110.385022	3.273923	9.384778	3.217829	13.642866	1.827423	0.076459	240.971915
min	0.000000	1.000000	0.000000	0.000000	19.400000	33.000000	991.200000	0.000000	0.000000
25%	0.000000	49.000000	0.000000	0.000000	21.900000	58.000000	993.800000	0.000000	0.000000
50%	33.500000	71.500000	0.075000	11.250000	23.850000	70.000000	995.100000	0.000000	0.100000
75%	67.500000	186.750000	3.505000	16.500000	26.075000	78.750000	996.400000	0.000000	214.475000
max	359.000000	356.000000	17.470000	35.700000	32.900000	87.000000	999.200000	0.800000	835.000000

3 Apéndice

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

Es un entorno muy interesante de programación, me recordó mucho a cuando en mi preparatoria usábamos un servidor local en el navegador para crear bases de datos dinámicas, solo que este entorno es mas limpio y en realidad se programa, sin mencionar que la librería panda es muy buena para el análisis de datos.

2. ¿Se te dificultó leer código en Python?

Al iniciar si se me dificulto un poco, ya que no tenia experiencia alguna con este lenguaje de programación. Pero revisando e investigando las distintas funciones que presenta panda y Python, además de la ayuda del profesor y compañeros, al realizar las preguntas respecto al estado y ciudad seleccionada, fue fácil.

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

Siento que los comandos e instrucciones de Fortran son mas claros en lo que hacen, pero la sintaxis y forma de trabajo de Python son mucho mejor. El lenguaje Fortran es de compilación, mientras que Python es de interpretación, lo cual siempre he encontrado (en mi opinión) que es mejor.

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

Graficar es mucho mas facil en Python que en Fortran. En Fortran se tenia que generar los datos, para posteriormente introducirlos a GNUPlot, y en veces este no hacia lo que se le pedia, a pesar de no tener errores al declarar la grafica. Python, en caso de tener un error, lo muestra y es claro en ello.

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

Es muy interesante y limpio, la facilidad de poder parar un proceso en caso de ser erróneo es algo grandioso, además de que con la experiencia anterior que tengo de programación, Python es de los lenguajes mas fáciles de manejar que he tenido la oportunidad de usar.

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

Al comenzar la actividad, pensé que era demasiado material nuevo, pero eso se debia a mi poca experiencia con el lenguaje. Una vez trabajando en ello me di cuenta que en realidad son los usos básicos de Python, análisis de datos y graficas. La practica fue una muy buena introducción al entorno de Python y su uso.

7. ¿Comentarios adicionales que desees compartir?

Esa practica fue una muy buena introducción al entorno de Python y sus funciones, ademas de su ambiente de Jupyter.

4 Bibliografía

- Kumar, Nikhil. Graph Plotting in Python. Recuperado de: www.geeksforgeeks.org/graph-plotting-in-python-set-1/
- Quintero, Francisco (2012, Septiembre 2). Lenguajes de Programación. Recuperado de: otroespacioblog.wordpress.com/2012/09/02/lenguajes-de-programacion-compilados-vs-interpretados/
- Python. (2018, Febrero 4). Recuperado de: https://en.wikipedia.org/wiki/Python_(programming_language)

Imagenes utilizadas:

- Simbolo Jupyter: Fecha: Agosto 2017, https://gitlab.eurecom.fr/zoe-apps/zapp-jupyter/blob/master/logo.png
- Simbolo Python: https://www.python.org/community/logos/