Reporte de Actividad 2

Diego Iván Moreno Campa

2 de Febrero, 2018

1 Introducción

Esta actividad es el segundo trabajo para la materia de Física computacional I en la licenciatura de Física.

El propósito de la actividad es familiarizarse con el entorno de Jupyter para escribir en lenguaje Python, el cual tiene numerosas funciones; estudiaremos las bibliotecas de Python para realizar cálculos numéricos Numpy y manejo de datos Pandas. La forma en que usaremos Python es para manejar datos de las estaciones en localidades Mexicanas.

2 Inicializando el entorno de desarrollo

Para empezar a utilizar Python tenemos que abrir el entorno en el que trabajaremos, Jupyter, por lo tanto, comenzamos abriendo una terminal localizada en el espacio de trabajo de nuestro ordenador y después ejecutamos el comando "*Jupyter Notebook*" en la terminal para iniciar el servidor de Jupyter en el directorio local y comenzar a escribr en la plataforma.

3 Búsqueda de datos

Para comenzar a practicar con Python, es necesario tener unos datos a manejar, es por eso que buscamos datos en la página que se nos proporcionó, donde se pueden encontrar los datos sobre el entorno de alguna posición de lectura del ambiente en México: "http://smn1.conagua.gob.mx/emas/".

Personalmente elegí nogales no solo porque esta cerca de casa sino porque también de los que había elegido anteriormente fue uno de los únicos que tenia datos limpios, consistentes y sin renglones vacíos.

En el documento de datos sobre el entorno

4 El Documento

Para crear el documento donde escribiremos el código en Python, se necesita crear una libreta en Jupyter entrando a la plataforma y seleccionando el listado desplegable "*New* ▼" y seleccionando la opción Python 3.0. El profesor nos proporcionó con un ejemplo para la organización de datos en Python dentro de su repositorio en GitHub.

4.1 Declarando Paquetes

La primera celda que utilizamos fue la declaración de paquetes para el archivo de Python, donde se declararon los siguientes paquetes:

- Pandas: Una librería que provee herramientas de alto rendimiento al analizar y estructurar datos para el lenguaje de programación Python.
- NumPy: Una librería para el lenguaje de programación Python, que añade soporte a arreglos y matríces multidimensionales de numeros grandes, así como también se proporciona una gran colección de funciones matemáticas de alto nivel para operar en estos arreglos.
- MatplotLib: Una librería para generar gráficas en Python

Los cuales se declararon como se muestra a continuación, con nombres cortos para después ser llamados ágilmente.

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

Para ejecutar una celda dentro del entorno Jupyter se utiliza el comando de teclado "*Shift+Enter*"

4.2 Entrada de datos

Continuamos en la siguiente celda introduciendo los datos de la hoja de texto con los datos del entorno de Nogales. Para esto declaramos una variable, en este caso llamada df0, que tomara los valores de la hoja de texto que le especifiquemos a través una función proporcionada por Pandas para leer archivos y traerlos a DataFrame, "read_ csv()". Se especificó la dirección del archivo que se lee, después se agregó un atributo para saltarse las primeras 4 filas, las cuales no contienen ningún dato manejable, y se agregó el atributo "sep='\s+'" para leer los datos con más de un espacio entre columnas.

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

4.3 Leer encabezado

Para leer el encabezado de un archivo de datos se utiliza la función head(), la cual imprime las primeras 5 filas del archivo.

```
df0.head()
```

	DD/MM/AAAA	HH:MM	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RAD-SOL
0	01/02/2018	16:30	96	112	7.0	10.2	13.2	24	877.0	0.0	495.0
1	01/02/2018	16:40	89	98	6.6	9.6	14.6	22	877.1	0.0	527.0
2	01/02/2018	16:50	104	94	5.8	8.2	15.4	21	877.2	0.0	560.0
3	01/02/2018	17:00	123	138	3.1	5.9	17.2	19	877.3	0.0	587.0
4	01/02/2018	17:10	82	116	3.8	6.9	18.7	18	877.3	0.0	614.0

4.4 Estructuramiento de datos

En estos momentos los datos estan desorganizados y no tienen forma dentro del contexto de Python, sin embargo, para esto existe la funcion DataFrame() de Pandas, que le da una estructura al texto para que pueda ser leído por el código. Declaramos una nueva variable que contenga la función Dataframe() con el atributo df0, la variable sin estructura que declaramos al inicio, y con el nombre df para abreviar.

Después, en la siguiente celda ejecutamos una función para observar el tipo de datos que puede leer Python para cada columna de datos.

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

```
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
RAD-SOL       float64
dtype: object
```

Para continuar con el estructuramiento de datos, en la siguiente celda se ejecuta una funcion sobre el DataFrame para combinar las columnas de "DD/MM/AAAA" y "HH:MM" en una sola columna llamada Fecha con el tipo Tiempo, como le sigue, se eliminan las correspondientes columnas que se combinaron. Observamos los cambios volviendo a imprimir el encabezado:

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

```
df.head()
```

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RAD-SOL	FECHA
0	96	112	7.0	10.2	13.2	24	877.0	0.0	495.0	2018-02-01 16:30:00
1	89	98	6.6	9.6	14.6	22	877.1	0.0	527.0	2018-02-01 16:40:00
2	104	94	5.8	8.2	15.4	21	877.2	0.0	560.0	2018-02-01 16:50:00
3	123	138	3.1	5.9	17.2	19	877.3	0.0	587.0	2018-02-01 17:00:00
4	82	116	3.8	6.9	18.7	18	877.3	0.0	614.0	2018-02-01 17:10:00

4.5 Resumen y selección de datos

Siempre que tenemos un listado o arreglo de datos es importante conocer los aspectos característicos de estos, como el promedio, la desviación y demás. En Python existe una función, describe(), para realizar esto sobre DataFrames, una función que imprime un resumen de dichas características del dataframe donde se utiliza. Esta función solo utiliza, claro está, tipos numéricos y no objetos como la fecha o la hora.

Es importante para el manejo de datos el tener una forma de seleccionar un intervalo de datos requeridos o deseados, no utilizamos una función explicita

```
df.describe()
```

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RAD-SOL
count	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.000000	143.0	143.000000
mean	211.748252	234.048951	5.908392	9.646154	15.965734	23.027972	877.748951	0.0	191.615385
std	80.572465	78.065008	2.885430	5.805349	5.690431	8.285355	1.404764	0.0	286.217560
min	17.000000	69.000000	1.300000	1.800000	8.800000	11.000000	875.300000	0.0	0.000000
25%	141.500000	203.000000	3.550000	4.900000	10.600000	14.000000	876.600000	0.0	0.000000
50%	245.000000	262.000000	5.500000	7.700000	14.600000	22.000000	878.400000	0.0	0.000000
75%	268.000000	290.500000	8.100000	14.100000	22.400000	31.000000	878.900000	0.0	424.500000
max	326.000000	359.000000	13.900000	27.000000	25.100000	36.000000	880.200000	0.0	797.000000

para realizar esto, sin embargo, se realizó fácilmente al declarar algunas variables que solo tomaban un intervalo temperatura de df.

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

	DIRS	DIRR	VELS	VELR	TEMP	HR	PB	PREC	RAD-SOL	FECHA
24	243	235	8.4	18.1	24.5	12	875.9	0.0	790.0	2018-02-01 20:30:00
25	211	311	6.3	14.0	24.5	11	875.8	0.0	757.0	2018-02-01 20:40:00
26	265	280	8.1	15.6	24.6	12	875.7	0.0	721.0	2018-02-01 20:50:00
27	252	300	7.4	14.0	24.7	11	875.6	0.0	716.0	2018-02-01 21:00:00
28	211	244	10.2	17.8	24.5	11	875.5	0.0	706.0	2018-02-01 21:10:00
29	205	250	7.7	23.8	24.7	11	875.4	0.0	690.0	2018-02-01 21:20:00
30	235	236	12.6	20.9	24.9	12	875.4	0.0	665.0	2018-02-01 21:30:00
33	246	276	8.4	16.6	24.8	13	875.4	0.0	453.0	2018-02-01 22:00:00
34	247	320	13.9	27.0	24.8	13	875.5	0.0	526.0	2018-02-01 22:10:00
35	302	289	8.2	20.8	24.4	13	875.5	0.0	474.0	2018-02-01 22:20:00
36	262	309	10.8	19.2	24.6	13	875.5	0.0	396.0	2018-02-01 22:30:00
37	267	286	10.1	16.9	24.4	13	875.4	0.0	317.0	2018-02-01 22:40:00
38	246	274	8.6	16.7	24.2	13	875.4	0.0	304.0	2018-02-01 22:50:00
39	254	236	6.5	12.5	24.3	13	875.4	0.0	307.0	2018-02-01 23:00:00
40	251	251	10.0	18.0	24.4	13	875.4	0.0	247.0	2018-02-01 23:10:00

Luego, si se desea únicamente conocer cierta característica estadística de los valores dados podemos usar ciertas funciones sobre el listado de datos, por ejemplo la funcion mean() que se puede usar sobre el DataFrame que tenemos, df. Y si deseamos obtener esta característica para una sola columna de datos entonces se especifíca ésta para mostrar únicamente este valor.

```
df.mean()
```

```
DIRS       211.748252
DIRR       234.048951
VELS         5.908392
VELR         9.646154
TEMP        15.965734
HR          23.027972
PB         877.748951
PREC         0.000000
RAD-SOL    191.615385
dtype: float64
```

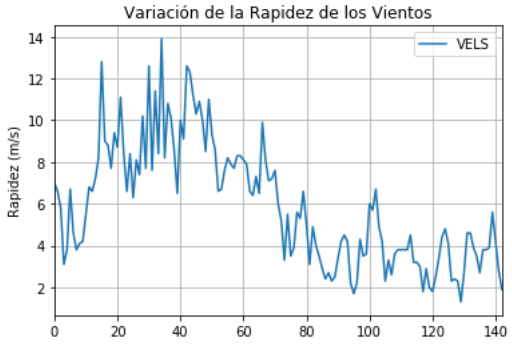
```
df.TEMP.mean()
```

```
15.965734265734264
```

4.6 Trazado de gráficas

Ahora podemos empezar a graficar nuestros datos estructurados, para esto primero llamamos a matplotlib *plt*, usando la función figure() para crear un objeto, este sera una gráfica de la rapidez de los vientos tomada de nuestra estructura de datos *df*. Después le agregamos el título, la etiqueta del "eje y", le agregamos la cuadrícula y se imprime con la funcion show().

```
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()
```

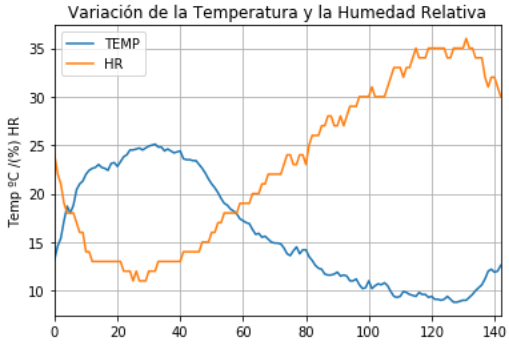


La siguiente gráfica que se realizó consistió en presentar dos columnas de datos en una sola gráfica, a modo de comparación. Esta vez como utilizaremos más de una columna de datos es mejor especifícar en otra variable las columnas de datos que tomaremos, por ejemplo como se realiza con la variable *df1*.

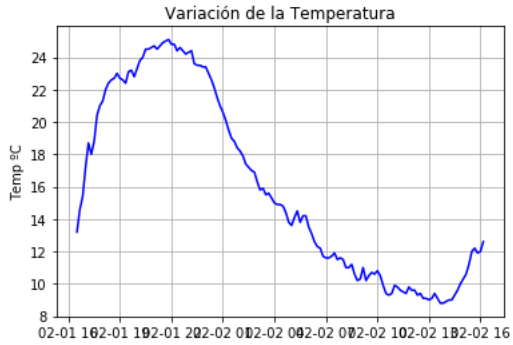
También podemos graficar alguna relación entre dos columnas de datos asignandole a cada eje una columna de datos:

```
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()
```

```
<matplotlib.figure.Figure at 0x7f885e0f1828>
```




```
plt.plot_date(x=df.FECHA, y=df.TEMP, fmt="b-")
plt.title("Variación de la Temperatura")
plt.ylabel("Temp ºC")
plt.grid(True)
plt.show()
```

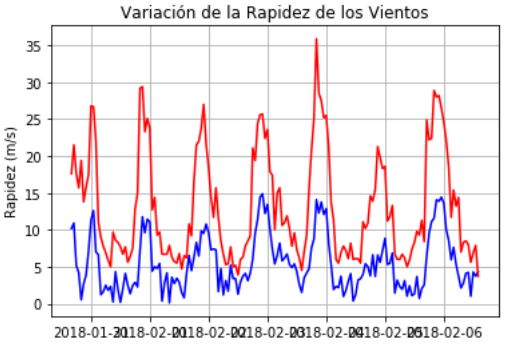


5 Actividades Adicionales

En la práctica se nos pidió realizar varias actividades después de haber escrito el ejemplo del profesor.

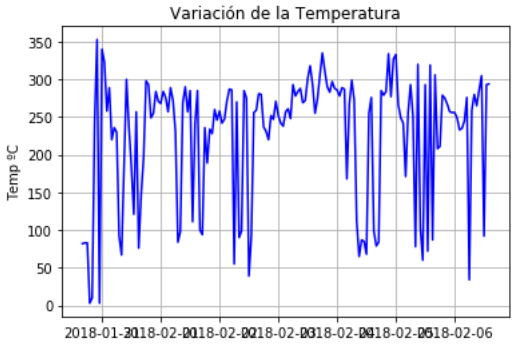
La primera actividad consiste en crear una gráfica que muestre la rapidez de los vientos y la rapidez de las ráfagas, como funciones del tiempo. La gráfica se llevó a cabo utilizando el mismo procedimiento que se observo en el ejemplo para gráficar una columna de datos con respecto a otra:

```
plt.plot_date(x=df.FECHA, y=df.VELS, fmt="b-")
plt.plot_date(x=df.FECHA, y=df.VELR, fmt="r-")
plt.title("Variación de la Rapidez de los Vientos")
plt.ylabel("Rapidez (m/s)")
plt.grid(True)
plt.show()
```



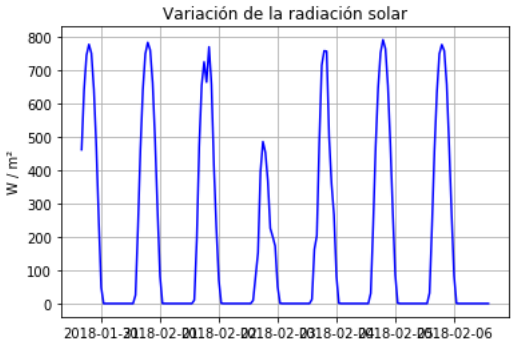
La segunda actividad consistió en crear una gráfica que muestre las direcciones, en unidades de grados, de los vientos con respecto al tiempo. Se puede notar claramente que los vientos son consistentes durante la noche hacia los 250 grados.

```
plt.plot_date(x=df.FECHA, y=df.DIRS, fmt="b-")
plt.title("Variación de la Dirección del viento")
plt.ylabel("Grados")
plt.grid(True)
plt.show()
```



La tercera actividad pedia crear una gráfica que muestre la radiación solar contra el tiempo. la cual se puede notar como la radiación solar es más intensa durante el medido día como se puede asumir lógicamente.

```
plt.plot_date(x=df.FECHA, y=df.RADSOL, fmt="b-")
plt.title("Variación de la radiación solar")
plt.ylabel("W / m²")
plt.grid(True)
plt.show()
```



La cuarta actividad actividad nos pide obtener el rango de temperatura diario de nuestros datos.

```
dftm = df.TEMP.max()-df.TEMP.min()
dftm
```

```
18.600000000000001
```

La quinta actividad se nos pide comentar sobre la relación entre la temperatura y la humedad, la cual me parece la temperatura y la humedad son inversamente proporcional una de la otra después de observar la gráfica donde se muestran juntas. Se puede deducir pensando en el concepto microscópico de temperatura y la humedad como la concentración de moleculas de agua en el aire: entre mayor es la temperatura, mayor energía habrá en el sistema, por lo que las moleculas tendran mayores velocidades y se separaran unas de las otras, lo que hará que las moleculas de agua esten menos concentradas, esto es, habra menor humedad debido a que el agua se evapora con la temperatura.

6 Bibliografía

- Numpy. (2018, Febrero 3) NumPy developers. Página Principal. http://www.numpy.org/
- Pandas Python Data Analysis. (2018, Febrero 3) Python developers. Página Principal. https://pandas.pydata.org/

Apéndice

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

   Parece un entorno de desarrollo fácil de manejar a pesar de que no se programar en Python

2. ¿Se te dificultó leer código en Python?

   Con mi experiencia de programador leer las funciones y variables no me pareció dificil de comprender lo que se realizaba excepto en algunos casos, y me parece que no a muchos se les dificulto esta parte

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

   De alguna manera me parece más fácil a la hora de organizar datos, sin embargo, como esta práctica no nos llevo muy a fondo al cálculo de tipos numéricos me parece dificil comparar ambos lenguajes en este aspecto

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

   Tener una forma directa de graficar tus datos me parece algo útil.

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

   Me pareció que en esta práctica no se pudo apreciar la funcionalidad de Python muy bien en cuanto a otros usos. Aprender Python no me parece de mucha utilidad en estos momentos debido a que manejo relativamente bien Fortran, si Python me demuestra que puede hacer lo que Fortran de manera portatil y rápida, pues todo bien.

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

   Definitivamente mucho material nuevo, no me gusta cuando me obligan a hacer algo que no tengo la menor idea de que esta haciendo al inicio. Me gusta saber lo que mi programa hará y que puedo hacer o con que puedo hacerlo.

7. ¿Comentarios adicionales que desees compartir?
   Ninguno mas de lo anterior