Carga de datos

In [237]:
```
import numpy as np
import pandas as pd
import csv
%matplotlib inline
import matplotlib.pyplot as plt
from pandas.plotting import register_matplotlib_converters
```

In [170]:
```
df = pd.read_csv('./estaciones_bici.csv', delimiter=';', encoding="utf-8")    #Archivo cargado en df
df
```

Out[170]:

	_id	available	connected	download_date	estation	free	open	ticket	total	updated
0	5c6050a42554172704fccdc0	9	1	2019-02-10 17:25:37.787	64	11	1	0	20	2019-02-10 17:21:13.000
1	5c6050a42554172704fccdc1	6	1	2019-02-10 17:25:37.787	73	14	1	1	20	2019-02-10 17:24:13.000
2	5c605be225541729b7d50885	20	1	2019-02-10 18:13:39.827	63	0	1	1	20	2019-02-10 18:09:16.000
3	5c605be225541729b7d50886	6	1	2019-02-10 18:13:39.827	64	14	1	0	20	2019-02-10 18:12:15.000
4	5c605be225541729b7d50887	9	1	2019-02-10 18:13:39.827	65	10	1	1	19	2019-02-10 18:09:16.000
...	...	...	...	...	...	...	...	...	...	...
27542	5c61face25541729b7d57419	0	1	2019-02-11 23:44:00.786	260	20	1	0	20	2019-02-11 23:42:16.000
27543	5c61face25541729b7d5741a	15	1	2019-02-11 23:44:00.786	261	4	1	0	19	2019-02-11 23:39:16.000
27544	5c61face25541729b7d5741b	1	1	2019-02-11 23:44:00.786	268	9	1	1	10	2019-02-11 23:42:16.000
27545	5c61face25541729b7d5741c	1	1	2019-02-11 23:44:00.786	269	14	1	0	15	2019-02-11 23:39:16.000
27546	5c61face25541729b7d5741d	8	1	2019-02-11 23:44:00.786	276	12	1	1	20	2019-02-11 23:42:16.000

27547 rows × 10 columns

Cargamos el archivo al dataframe "df" el cual usaremos a lo largo del documento, se utiliza la librería panda para hacer la conversión del archivo en formato csv a dataframe.

In [172]:
```
df = df.rename(columns={"_id":"ID","available":"Disponible","connected":"Conectado","download_date":"Fecha Descarga","estation":'
df
```

Out[172]:

	ID	Disponible	Conectado	Fecha Descarga	Estacion	Libre	Abierto	Tiquete	Total	Actualizado
0	5c6050a42554172704fccdc0	9	1	2019-02-10 17:25:37.787	64	11	1	0	20	2019-02-10 17:21:13.000
1	5c6050a42554172704fccdc1	6	1	2019-02-10 17:25:37.787	73	14	1	1	20	2019-02-10 17:24:13.000
2	5c605be225541729b7d50885	20	1	2019-02-10 18:13:39.827	63	0	1	1	20	2019-02-10 18:09:16.000
3	5c605be225541729b7d50886	6	1	2019-02-10 18:13:39.827	64	14	1	0	20	2019-02-10 18:12:15.000
4	5c605be225541729b7d50887	9	1	2019-02-10 18:13:39.827	65	10	1	1	19	2019-02-10 18:09:16.000
...	...	...	...	...	...	...	...	...	...	...
27542	5c61face25541729b7d57419	0	1	2019-02-11 23:44:00.786	260	20	1	0	20	2019-02-11 23:42:16.000
27543	5c61face25541729b7d5741a	15	1	2019-02-11 23:44:00.786	261	4	1	0	19	2019-02-11 23:39:16.000
27544	5c61face25541729b7d5741b	1	1	2019-02-11 23:44:00.786	268	9	1	1	10	2019-02-11 23:42:16.000
27545	5c61face25541729b7d5741c	1	1	2019-02-11 23:44:00.786	269	14	1	0	15	2019-02-11 23:39:16.000
27546	5c61face25541729b7d5741d	8	1	2019-02-11 23:44:00.786	276	12	1	1	20	2019-02-11 23:42:16.000

27547 rows × 10 columns

En mi caso cambie los nombres de las columnas para hacer mas entendible el manejo de los datos, viendo los datos podemos quitar el ID y la Fecha de Descarga y quedarnos solamente con Disponible, Conectado, Estación, Libre, Abierto, Tiquete, Total y Actualizado.

Obtención de número de estaciones con total de 30

Para encontrar las estaciones que tienen una capacidad Total de 30, realizamos un filtro de la columna Total del dataframe y lo guardamos en una variable que nos va indicar cual fila cumple (True y cual no (False con el criterio de filtrado, luego simplemente lo mostramos o podemos usar la función "len", esto nos da el resultado de 1098 Estaciones con una capacidad de 30

```
In [173]: estaciones_30 = df['Total'] == 30
          df[estaciones_30]
```

Out[173]:

	ID	Disponible	Conectado	Fecha Descarga	Estacion	Libre	Abierto	Tiquete	Total	Actualizado
29	5c605be225541729b7d508a0	30	1	2019-02-10 18:13:39.827	101	0	1	1	30	2019-02-10 18:09:16.000
34	5c605be225541729b7d508a5	19	1	2019-02-10 18:13:39.827	117	11	1	1	30	2019-02-10 18:12:15.000
82	5c605be225541729b7d508d5	7	1	2019-02-10 18:13:39.827	89	23	1	0	30	2019-02-10 18:12:15.000
86	5c605be225541729b7d508d9	11	1	2019-02-10 18:13:39.827	105	19	1	0	30	2019-02-10 18:12:15.000
113	5c605be225541729b7d508f4	21	1	2019-02-10 18:13:39.827	143	8	1	1	30	2019-02-10 18:09:16.000
...	...	...	...	...	...	...	...	...	...	...
27417	5c61face25541729b7d5739c	0	1	2019-02-11 23:44:00.786	111	30	1	0	30	2019-02-11 23:39:16.000
27419	5c61face25541729b7d5739e	0	1	2019-02-11 23:44:00.786	114	30	1	0	30	2019-02-11 23:39:16.000
27460	5c61face25541729b7d573c7	9	1	2019-02-11 23:44:00.786	189	21	1	1	30	2019-02-11 23:39:16.000
27490	5c61face25541729b7d573e5	1	1	2019-02-11 23:44:00.786	226	29	1	0	30	2019-02-11 23:39:16.000
27510	5c61face25541729b7d573f9	0	1	2019-02-11 23:44:00.786	246	20	1	0	30	2019-02-11 23:39:16.000

1098 rows × 10 columns

```
In [358]: len (df[estaciones_30]) #ESTACIONES CON CAPACIDAD DE 30 SON 1098.
Out[358]: 1098
```

Número de estación con la media más alta de bicis disponibles

Para encontrar la estación con la media mas alta disponible, tenemos primero que analizar el dataframe, viéndolo rápidamente notamos que hay alrededor de 100 filas por cada estación distribuido en todo el dataframe, por lo cual primero debemos agrupar el Dataframe por Estacion y luego realizar el cálculo de la media en la columna de "Disponible", para eso usamos la función "groupby" y "aggregate".

```
In [359]: estacion_grupo = df.groupby('Estacion').aggregate({'Disponible':['mean', 'count']})
          estacion_grupo
```

Out[359]:

	Disponible	
	mean	count
Estacion		
1	12.040000	100
2	2.070000	100
3	3.300000	100
4	3.100000	100
5	8.730000	100
...	...	...
272	13.444444	99
273	6.242424	99
274	4.989899	99
275	16.222222	99
276	15.323232	99

276 rows × 2 columns

```
In [360]: solo_media = estacion_grupo.iloc[:, 0]
```

```
In [361]: media = solo_media.sort_values(0, ascending=False)   #La estacion con la media mas alta de bicis Disponible es la #50
```

```
In [367]: media_alta = media.iloc[0:1]
          media_alta
```

```
Out[367]: Estacion
          50    31.26
          Name: (Disponible, mean), dtype: float64
```

Una ves que tenemos el DataFrame con el número de estación y la media calculada para cada una, simplemente tenemos que usar la función "sort" con el operando "ascending=False" para que nos organice los datos de mayor a menor y luego con la función "iloc" escogemos la fila 1 que va ser el que tiene la media mas alta de Disponible y luego solo imprimimos el valor para saber cual es, que en este caso es la Estación 50 con una media de 31.26.

Realizar el histograma de la estación de bicis disponibles de la Estación 50

Lo primero es extraer del dataframe principal, los datos de la Estación 50, para luego trabajar sobre ellos únicamente.

```
In [210]: filtro_50 = df['Estacion'] == 50     #Filtramos la estacion 50 y guardamos el dataframe en una variable
          estacion_50 = df[filtro_50]
          estacion_50
```

Out[210]:

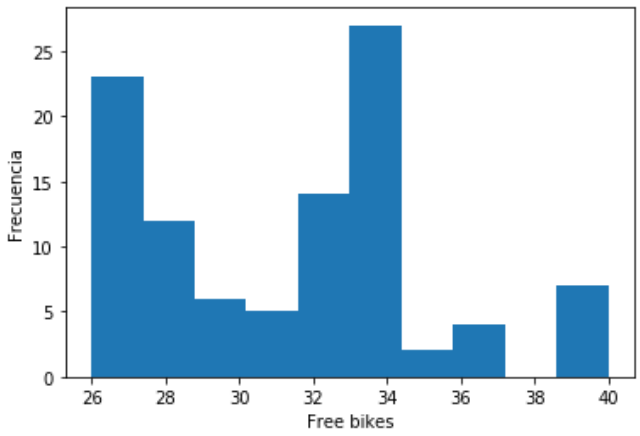
	ID	Disponible	Conectado	Fecha Descarga	Estacion	Libre	Abierto	Tiquete	Total	Actualizado
74	5c605be225541729b7d508cd	39	1	2019-02-10 18:13:39.827	50	1	1	1	40	2019-02-10 18:12:15.000
350	5c605f8625541729b7d509e1	36	1	2019-02-10 18:29:13.135	50	4	1	1	40	2019-02-10 18:24:14.000
625	5c60632925541729b7d50af4	33	1	2019-02-10 18:44:43.728	50	7	1	1	40	2019-02-10 18:42:14.000
901	5c6066cf25541729b7d50c08	31	1	2019-02-10 19:00:14.475	50	9	1	1	40	2019-02-10 19:00:14.000
1177	5c606a7325541729b7d50d1c	33	1	2019-02-10 19:15:48.800	50	7	1	1	40	2019-02-10 19:15:16.000
...	...	...	...	...	...	...	...	...	...	...
26239	5c61ec4225541729b7d56f02	32	1	2019-02-11 22:41:56.937	50	8	1	1	40	2019-02-11 22:39:17.000
26515	5c61efe525541729b7d57016	33	1	2019-02-11 22:57:27.760	50	7	1	1	40	2019-02-11 22:57:17.000
26791	5c61f38825541729b7d5712a	33	1	2019-02-11 23:12:58.602	50	7	1	1	40	2019-02-11 23:12:16.000
27067	5c61f72b25541729b7d5723e	32	1	2019-02-11 23:28:29.903	50	8	1	1	40	2019-02-11 23:27:14.000
27343	5c61face25541729b7d57352	32	1	2019-02-11 23:44:00.786	50	8	1	1	40	2019-02-11 23:42:15.000

100 rows × 10 columns

Ya con el dataframe de la estación 50 guardado podemos usar la librería matplotlib, para plotear el histograma en base a la columna de "Disponible"

```
In [221]: plt.hist(estacion_50['Disponible'])
          plt.ylabel('Frecuencia')
          plt.xlabel('Free bikes')
```

Out[221]: Text(0.5, 0, 'Free bikes')



Realizar gráfica con la línea temporal de bicis disponibles de la Estación 50

Como para el histrograma ya teníamos el dataframe de la estación 50 filtrado, usamos la misma variable para trabajar el grafico lineal.

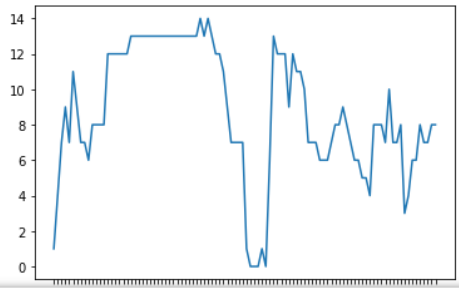
Establecemos y extraemos el eje Y (Data) y el eje X (Time), que respectivamente corresponden a las columnas de "Libre" y "Actualizado" del DataFrame.

Luego creamos un nuevo DataFrame, con únicamente las dos columnas que queremos graficar Time y Data, el DataFrame cuando lo creamos debemos usar la función de transpuesta para cambiar columnas por filas.

Una ves tenemos el DataFrame (df_T) podemos plotear usando la librearia matplotlib.

```
In [366]: # Ordenar la serie temporal
          gl_50 = estacion_50.sort_values('Actualizado', ascending=True)
          time = gl_50['Actualizado']
          data = gl_50['Libre']
          df_lineal = pd.DataFrame([time, data])
          df_T = df_lineal.T
          df_T.columns = ["Time", "Data"]
          df_T = df_T.sort_values('Time', ascending=True)
          plt.plot(df_T['Time'], df_T['Data'])
          plt.xticks(rotation='vertical')
```

```
 96,
 97,
 98,
 99],
<a list of 100 Text xticklabel objects>)
```



Nota: Una problemática que no pude resolver es que las fechas salieran por rangos en el eje X, como está el código se imprimen todas las fechas del DataFrame en el eje X y lo satura y por lo tanto no se ve bien. Si pudiera tener un feedback de como resolver este problema porque experimente y probé de todo, gracias.