mcpp_taller6_Julian_Ramirez

September 27, 2019

1 Taller 6

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 27-sep-2019 11:59 PM

Julián Santiago Ramírez
julians.ramirez@urosario.edu.co

1.1 Instrucciones:

- Guarde una copia de este *Jupyter Notebook* en su computador, idealmente en una carpeta destinada al material del curso.
- Modifique el nombre del archivo del *notebook*, agregando al final un guión inferior y su nombre y apellido, separados estos últimos por otro guión inferior. Por ejemplo, mi *notebook* se llamaría: mcpp_taller6_santiago_matallana
- Marque el *notebook* con su nombre y e-mail en el bloque verde arriba. Reemplace el texto "[Su nombre acá]" con su nombre y apellido. Similar para su e-mail.
- Desarrolle la totalidad del taller sobre este *notebook*, insertando las celdas que sea necesario debajo de cada pregunta. Haga buen uso de las celdas para código y de las celdas tipo *markdown* según el caso.
- Recuerde salvar periódicamente sus avances.
- Cuando termine el taller:

    1. Descárguelo en PDF. Si tiene algún problema con la conversión, descárguelo en HTML.
    2. Suba todos los archivos a su repositorio en GitHub, en una carpeta destinada exclusivamente para este taller, antes de la fecha y hora límites.

(Todos los ejercicios tienen el mismo valor.)

1.1.1 Resuelva la parte 1 de este documento.

```
In [91]: import numpy as np
         import scipy.linalg as la
         import matplotlib.pyplot as plt
         import math
         import pandas as pd
         import datetime
         from random import randint as ran
```

```
from matplotlib.dates import DateFormatter
import matplotlib.dates as mdates
```

1. Choose a value and set the variable x to that value.

```
In [4]: # Creamos la variable x
        x=20
```

2. What is command to compute the square of x? Its cube?

```
In [6]: ## Creamos el cuadrado
        cuadrado=x**2
        print("x al cuadrado: ",cuadrado)
        cubo=x**3
        print("x al cubo: ",cubo)
```

```
x al cuadrado:  400
x al cubo:  8000
```

3. Choose an angle and set the variable theta to its value (a number).

```
In [15]: theta=0.42354751
         print(theta)
```

```
0.42354751
```

4. What is sin ? cos ? Angles can be measured in degrees or radians. Which of these are being used used?

```
In [22]: seno=math.sin(theta)
         coseno=math.cos(theta)

         print("seno: ",seno)
         print("coseno: ",coseno)

         print("La libreria por defecto trabaja el seno y coseno con grados en radianes")
```

```
seno:  0.41099707261738
coseno:  0.9116366635342965
La libreria por defecto trabaja el seno y coseno con grados en radianes
```

5. Use the np.linspace function to create a row vector called meshPoints containing exactly 500 values with values evenly spaced between -1 and 1.

```
In [24]: ## Usamos la libreria numpy para crear el array
         meshPoints = np.linspace(-1, 1, 500)
         print(meshPoints)
```

```
[-1.         -0.99599198 -0.99198397 -0.98797595 -0.98396794 -0.97995992
 -0.9759519  -0.97194389 -0.96793587 -0.96392786 -0.95991984 -0.95591182
 -0.95190381 -0.94789579 -0.94388778 -0.93987976 -0.93587174 -0.93186373
 -0.92785571 -0.9238477  -0.91983968 -0.91583166 -0.91182365 -0.90781563
 -0.90380762 -0.8997996  -0.89579158 -0.89178357 -0.88777555 -0.88376754
 -0.87975952 -0.8757515  -0.87174349 -0.86773547 -0.86372745 -0.85971944
 -0.85571142 -0.85170341 -0.84769539 -0.84368737 -0.83967936 -0.83567134
 -0.83166333 -0.82765531 -0.82364729 -0.81963928 -0.81563126 -0.81162325
 -0.80761523 -0.80360721 -0.7995992  -0.79559118 -0.79158317 -0.78757515
 -0.78356713 -0.77955912 -0.7755511  -0.77154309 -0.76753507 -0.76352705
 -0.75951904 -0.75551102 -0.75150301 -0.74749499 -0.74348697 -0.73947896
 -0.73547094 -0.73146293 -0.72745491 -0.72344689 -0.71943888 -0.71543086
 -0.71142285 -0.70741483 -0.70340681 -0.6993988  -0.69539078 -0.69138277
 -0.68737475 -0.68336673 -0.67935872 -0.6753507  -0.67134269 -0.66733467
 -0.66332665 -0.65931864 -0.65531062 -0.65130261 -0.64729459 -0.64328657
 -0.63927856 -0.63527054 -0.63126253 -0.62725451 -0.62324649 -0.61923848
 -0.61523046 -0.61122244 -0.60721443 -0.60320641 -0.5991984  -0.59519038
 -0.59118236 -0.58717435 -0.58316633 -0.57915832 -0.5751503  -0.57114228
 -0.56713427 -0.56312625 -0.55911824 -0.55511022 -0.5511022  -0.54709419
 -0.54308617 -0.53907816 -0.53507014 -0.53106212 -0.52705411 -0.52304609
 -0.51903808 -0.51503006 -0.51102204 -0.50701403 -0.50300601 -0.498998
 -0.49498998 -0.49098196 -0.48697395 -0.48296593 -0.47895792 -0.4749499
 -0.47094188 -0.46693387 -0.46292585 -0.45891784 -0.45490982 -0.4509018
 -0.44689379 -0.44288577 -0.43887776 -0.43486974 -0.43086172 -0.42685371
 -0.42284569 -0.41883768 -0.41482966 -0.41082164 -0.40681363 -0.40280561
 -0.3987976  -0.39478958 -0.39078156 -0.38677355 -0.38276553 -0.37875752
 -0.3747495  -0.37074148 -0.36673347 -0.36272545 -0.35871743 -0.35470942
 -0.3507014  -0.34669339 -0.34268537 -0.33867735 -0.33466934 -0.33066132
 -0.32665331 -0.32264529 -0.31863727 -0.31462926 -0.31062124 -0.30661323
 -0.30260521 -0.29859719 -0.29458918 -0.29058116 -0.28657315 -0.28256513
 -0.27855711 -0.2745491  -0.27054108 -0.26653307 -0.26252505 -0.25851703
 -0.25450902 -0.250501   -0.24649299 -0.24248497 -0.23847695 -0.23446894
 -0.23046092 -0.22645291 -0.22244489 -0.21843687 -0.21442886 -0.21042084
 -0.20641283 -0.20240481 -0.19839679 -0.19438878 -0.19038076 -0.18637275
 -0.18236473 -0.17835671 -0.1743487  -0.17034068 -0.16633267 -0.16232465
 -0.15831663 -0.15430862 -0.1503006  -0.14629259 -0.14228457 -0.13827655
 -0.13426854 -0.13026052 -0.12625251 -0.12224449 -0.11823647 -0.11422846
 -0.11022044 -0.10621242 -0.10220441 -0.09819639 -0.09418838 -0.09018036
 -0.08617234 -0.08216433 -0.07815631 -0.0741483  -0.07014028 -0.06613226
 -0.06212425 -0.05811623 -0.05410822 -0.0501002  -0.04609218 -0.04208417
 -0.03807615 -0.03406814 -0.03006012 -0.0260521  -0.02204409 -0.01803607
 -0.01402806 -0.01002004 -0.00601202 -0.00200401  0.00200401  0.00601202
  0.01002004  0.01402806  0.01803607  0.02204409  0.0260521   0.03006012
  0.03406814  0.03807615  0.04208417  0.04609218  0.0501002   0.05410822
  0.05811623  0.06212425  0.06613226  0.07014028  0.0741483   0.07815631
  0.08216433  0.08617234  0.09018036  0.09418838  0.09819639  0.10220441
  0.10621242  0.11022044  0.11422846  0.11823647  0.12224449  0.12625251
  0.13026052  0.13426854  0.13827655  0.14228457  0.14629259  0.1503006
```

```
 0.15430862  0.15831663  0.16232465  0.16633267  0.17034068  0.1743487
 0.17835671  0.18236473  0.18637275  0.19038076  0.19438878  0.19839679
 0.20240481  0.20641283  0.21042084  0.21442886  0.21843687  0.22244489
 0.22645291  0.23046092  0.23446894  0.23847695  0.24248497  0.24649299
 0.250501    0.25450902  0.25851703  0.26252505  0.26653307  0.27054108
 0.2745491   0.27855711  0.28256513  0.28657315  0.29058116  0.29458918
 0.29859719  0.30260521  0.30661323  0.31062124  0.31462926  0.31863727
 0.32264529  0.32665331  0.33066132  0.33466934  0.33867735  0.34268537
 0.34669339  0.3507014   0.35470942  0.35871743  0.36272545  0.36673347
 0.37074148  0.3747495   0.37875752  0.38276553  0.38677355  0.39078156
 0.39478958  0.3987976   0.40280561  0.40681363  0.41082164  0.41482966
 0.41883768  0.42284569  0.42685371  0.43086172  0.43486974  0.43887776
 0.44288577  0.44689379  0.4509018   0.45490982  0.45891784  0.46292585
 0.46693387  0.47094188  0.4749499   0.47895792  0.48296593  0.48697395
 0.49098196  0.49498998  0.498998    0.50300601  0.50701403  0.51102204
 0.51503006  0.51903808  0.52304609  0.52705411  0.53106212  0.53507014
 0.53907816  0.54308617  0.54709419  0.5511022   0.55511022  0.55911824
 0.56312625  0.56713427  0.57114228  0.5751503   0.57915832  0.58316633
 0.58717435  0.59118236  0.59519038  0.5991984   0.60320641  0.60721443
 0.61122244  0.61523046  0.61923848  0.62324649  0.62725451  0.63126253
 0.63527054  0.63927856  0.64328657  0.64729459  0.65130261  0.65531062
 0.65931864  0.66332665  0.66733467  0.67134269  0.6753507   0.67935872
 0.68336673  0.68737475  0.69138277  0.69539078  0.6993988   0.70340681
 0.70741483  0.71142285  0.71543086  0.71943888  0.72344689  0.72745491
 0.73146293  0.73547094  0.73947896  0.74348697  0.74749499  0.75150301
 0.75551102  0.75951904  0.76352705  0.76753507  0.77154309  0.7755511
 0.77955912  0.78356713  0.78757515  0.79158317  0.79559118  0.7995992
 0.80360721  0.80761523  0.81162325  0.81563126  0.81963928  0.82364729
 0.82765531  0.83166333  0.83567134  0.83967936  0.84368737  0.84769539
 0.85170341  0.85571142  0.85971944  0.86372745  0.86773547  0.87174349
 0.8757515   0.87975952  0.88376754  0.88777555  0.89178357  0.89579158
 0.8997996   0.90380762  0.90781563  0.91182365  0.91583166  0.91983968
 0.9238477   0.92785571  0.93186373  0.93587174  0.93987976  0.94388778
 0.94789579  0.95190381  0.95591182  0.95991984  0.96392786  0.96793587
 0.97194389  0.9759519   0.97995992  0.98396794  0.98797595  0.99198397
 0.99599198  1.        ]
```

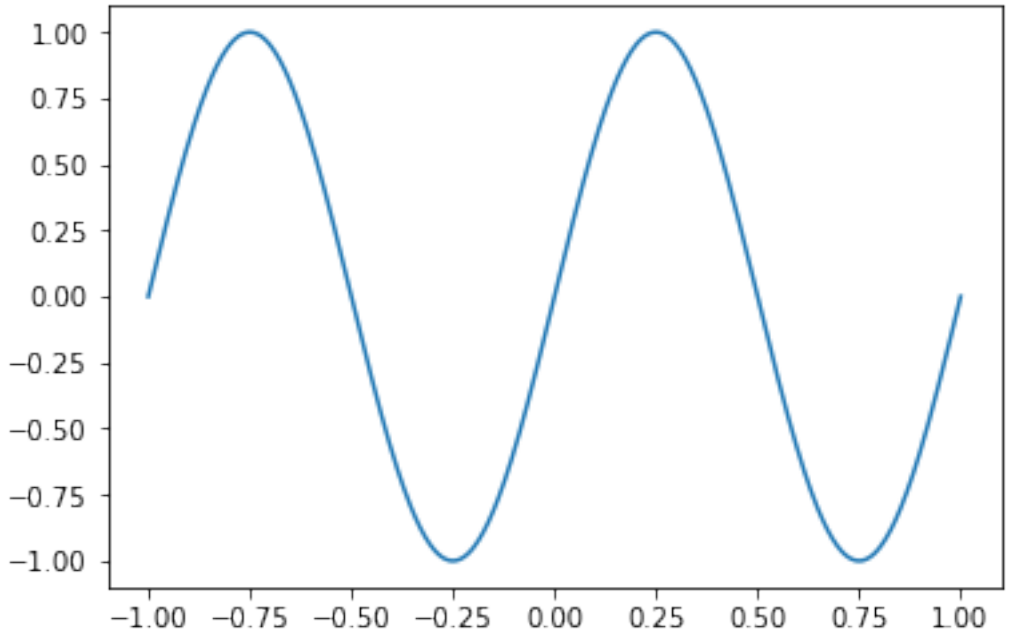
6. What expression will yield the value of the 53th element of meshPoints? What is this value?

```
In [25]: # Al ser un arreglo simplemente buscamos el elemento en la posicion 53 y como el arra
         dato=meshPoints[52]
         print("Valor del elemento 53: ",dato)
```

```
Valor del elemento 53:  -0.7915831663326653
```

7. Produce a plot of a sinusoid on the interval [1, 1] using the command plt.plot(meshPoints,np.sin(2*pi*meshPoints)) Please save this plot as a jpeg (.jpg) file and send it along with your work

```
In [27]: ### Graficamos el plot solicitado
         plt.plot(meshPoints, np.sin(2 * math.pi * meshPoints))
         plt.show()

         # Guardamos el archivo
         plt.savefig("jpeg.jpg")
```

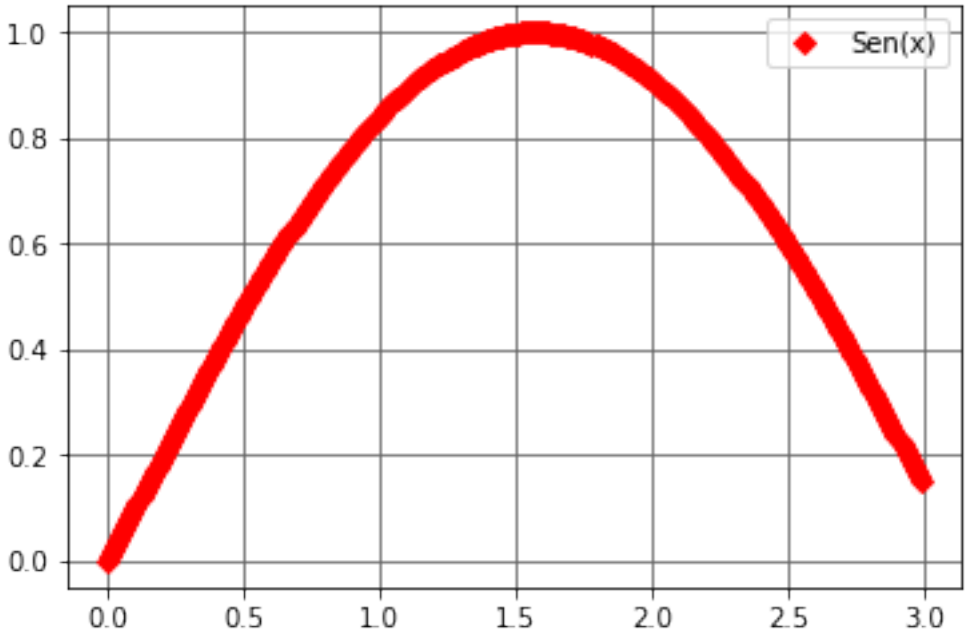



```
<Figure size 432x288 with 0 Axes>
```

1.1.2 Resuelva los ejercicios de las secciones 4.1, 5.1, 6.1, 7.4 y 8.5 de este documento.

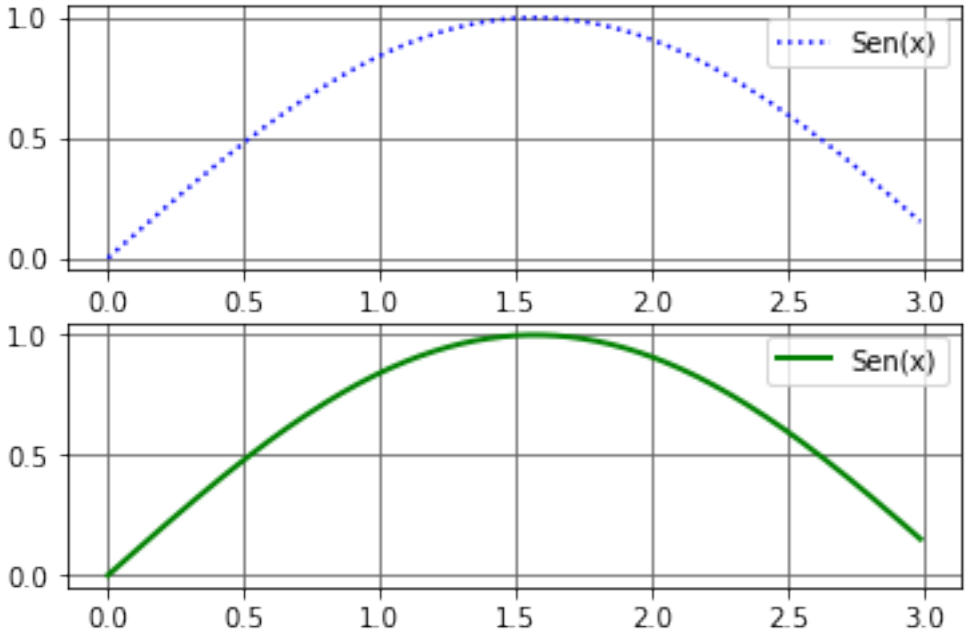
4.1 Exercises 1. Plot a simple graph of a sinus function in the range 0 to 3 with a step size of 0.01. 2. Make the line red. Add diamond-shaped markers with size of 5. 3. Add a legend and a grid to the plot.

```
In [42]: ## Rango en el eje x
         eje_x = np.arange(0, 3, 0.01);
         ## Sin (x) es la función solicitada
         eje_y = np.sin(eje_x);
         ## Plot
         plt.plot(eje_x, eje_y,'D',color = 'red',markersize  = 5)
         plt.gca().legend(('Sen(x)','y1'))
         plt.grid(b=True, which='major', color='#666666', linestyle='-')
```



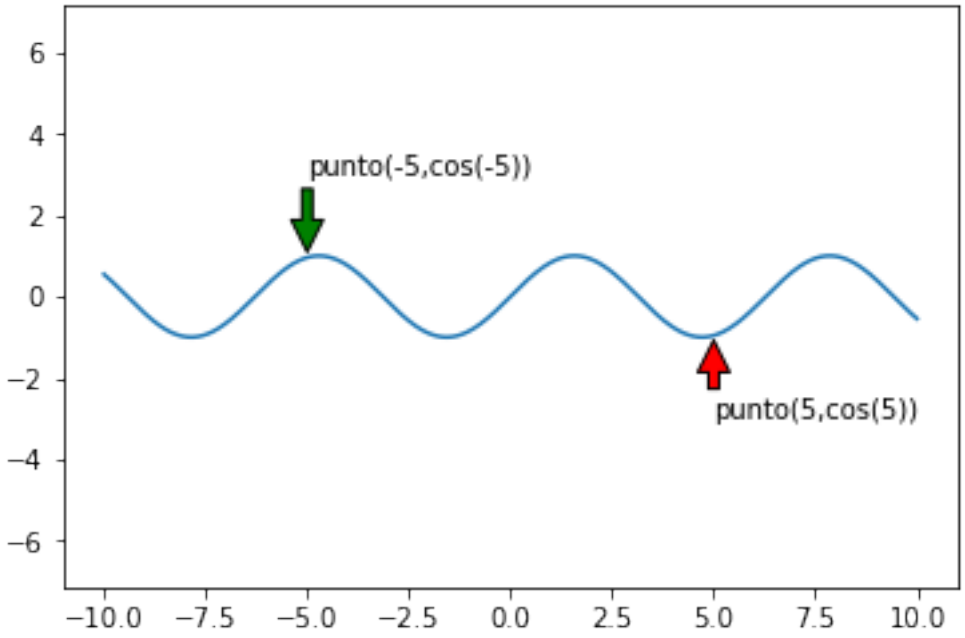
5.1 Exercise 1. Apply different line styles to a plot. Change line color and thickness as well as the size and the kind of the marker. Experiment with different styles.

```
In [55]: ## Rango en el eje x
         eje_x = np.arange(0, 3, 0.01);
         ## Sin (x) es la función solicitada
         eje_y = np.sin(eje_x);
         ## Plot
         plt.subplot(2, 1, 1)
         plt.plot(eje_x, eje_y,':',color = 'blue',markersize  = 1)
         plt.gca().legend(('Sen(x)','y1'))
         plt.grid(b=True, which='major', color='#666666', linestyle='-')

         plt.subplot(2, 1, 2)
         plt.plot(eje_x, eje_y,'r',color = 'green',markersize  = 6,linewidth  = 2)
         plt.gca().legend(('Sen(x)','y1'))
         plt.grid(b=True, which='major', color='#666666', linestyle='-')
```



6.1 Exercise 1. Annotate a line at two places with text. Use green and red arrows and align it according to figure points and data.

```
In [67]: ### Nececitamos definir los ax de la figura para agregar texto en el grafica
         fig, ax = plt.subplots()
         ### Entre mas grande sea el 3 dato del linspace sera mas nitida la grafica
         eje_x = np.linspace(-10,10,10000)
         ax.plot(eje_x, np.sin(eje_x))
         ax.axis('equal')
         ax.annotate('punto(-5,cos(-5))', xy=(-5, 1), xytext=(-5, 3),arrowprops=dict(facecolor=

         ax.annotate('punto(5,cos(5))', xy=(5,-1), xytext=(5, -3),arrowprops=dict(facecolor =
```



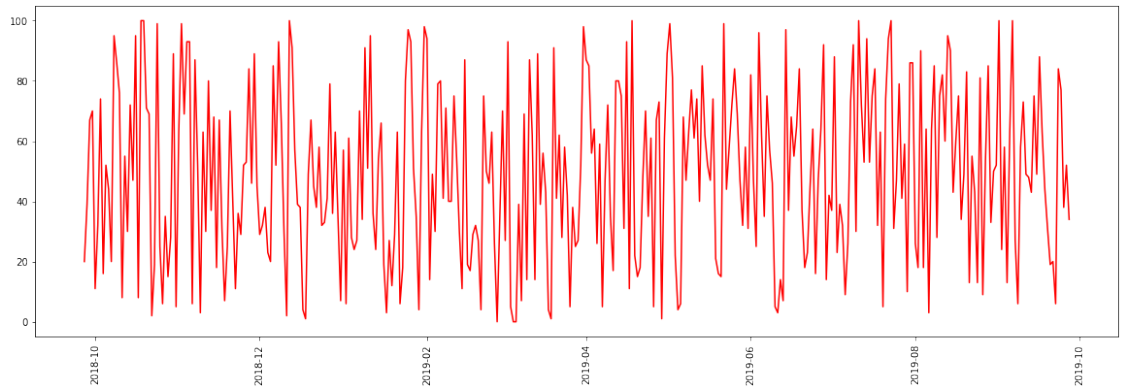
7.4 Exercises 1. Plot a graph with dates for one year with daily values at the x axis using the built-in module datetime. 2. Format the dates in such a way that only the first day of the month is shown. 3. Display the dates with and without the year. Show the month as number and as first three letters of the month name.

```
In [80]: ## Empezamos el dia 2019-01-01
         dia_inicial = datetime.datetime(2018,9,27)
         ## Cantidad dias año (son 365 pero para la crear la lista necesitamos un tope de 366)
         dias_año = 366
         ## Crearemos Una lista de los dias desde 2018-27-09 a 2019-27-09
         dias = []
         ## Tambien crearemos unos valores que almacenaremos en una lista de forma aleatorioa
         valores=[]
         for i in range (0, dias_año):
             dia_siguiente = dia_inicial + datetime.timedelta(days = i)
             dia_siguiente = dia_siguiente.strftime("%Y-%m-%d")
             dias.append(dia_siguiente)
             valores.append(ran(0,100))

         # Se muestra todo en una tabla de la siguiente forma
         tabla = pd.DataFrame()
         tabla['valor'] = valores
         tabla = tabla.set_index(pd.to_datetime(dias))
         print(tabla)
         #dateList = pd.to_datetime(dateList)
         #dateList
```

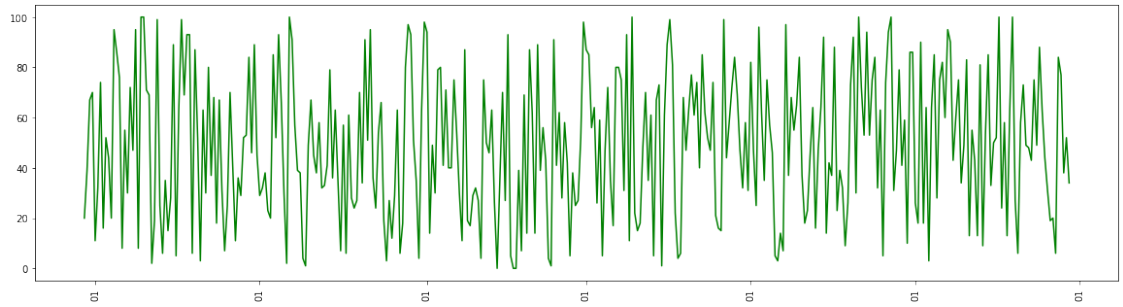
	valor
2018-09-27	20
2018-09-28	39
2018-09-29	67
2018-09-30	70
2018-10-01	11
2018-10-02	32
2018-10-03	74
2018-10-04	16
2018-10-05	52
2018-10-06	44
2018-10-07	20
2018-10-08	95
2018-10-09	86
2018-10-10	76
2018-10-11	8
2018-10-12	55
2018-10-13	30
2018-10-14	72
2018-10-15	47
2018-10-16	95
2018-10-17	8
2018-10-18	100
2018-10-19	100
2018-10-20	71
2018-10-21	69
2018-10-22	2
2018-10-23	19
2018-10-24	99
2018-10-25	25
2018-10-26	6
...	...
2019-08-29	33
2019-08-30	50
2019-08-31	52
2019-09-01	100
2019-09-02	24
2019-09-03	58
2019-09-04	13
2019-09-05	63
2019-09-06	100
2019-09-07	27
2019-09-08	6
2019-09-09	58
2019-09-10	73
2019-09-11	49
2019-09-12	48
2019-09-13	43

```
2019-09-14     75
2019-09-15     49
2019-09-16     88
2019-09-17     63
2019-09-18     44
2019-09-19     31
2019-09-20     19
2019-09-21     20
2019-09-22      6
2019-09-23     84
2019-09-24     77
2019-09-25     38
2019-09-26     52
2019-09-27     34

[366 rows x 1 columns]
```

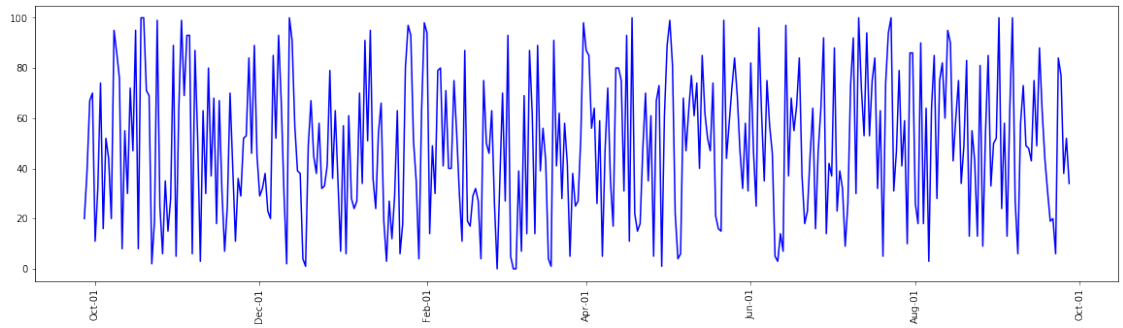
```
In [87]: fig, ax = plt.subplots()
         fig.subplots_adjust(bottom=0.09)
         plt.xticks(rotation=90)
         plt.plot(tabla,"red")
         plt.rcParams["figure.figsize"] = [20.0, 5.0]
```




```
In [93]: fig, ax = plt.subplots()
         fig.subplots_adjust(bottom=0.09)
         plt.xticks(rotation=90)
         plt.plot(tabla,"green")
         plt.rcParams["figure.figsize"] = [20.0, 5.0]
         ax.xaxis.set_major_formatter(DateFormatter('%d'))
```



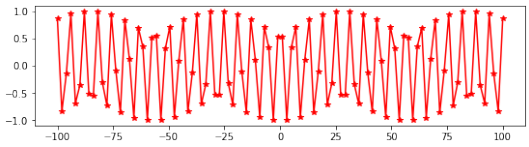

```
In [96]: fig, ax = plt.subplots()
         fig.subplots_adjust(bottom=0.09)
         plt.xticks(rotation=90)
         plt.plot(tabla,"blue")
         plt.rcParams["figure.figsize"] = [20.0, 5.0]
         ax.xaxis.set_major_formatter(DateFormatter('%b-%d'))
```

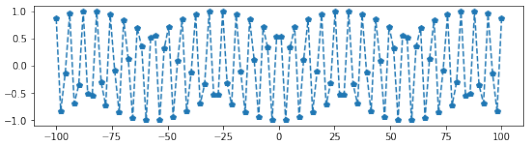


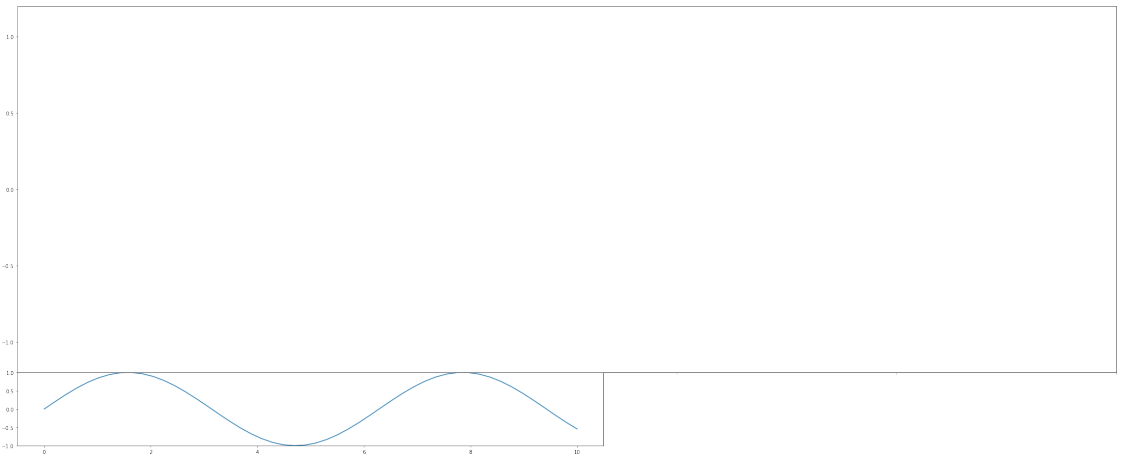
8.5 Exercises 1. Draw two figures, one 5 by 5, one 10 by 10 inches. 2. Add four subplots to one figure. Add labels and ticks only to the outermost axes. 3. Place a small plot in one bigger plot.

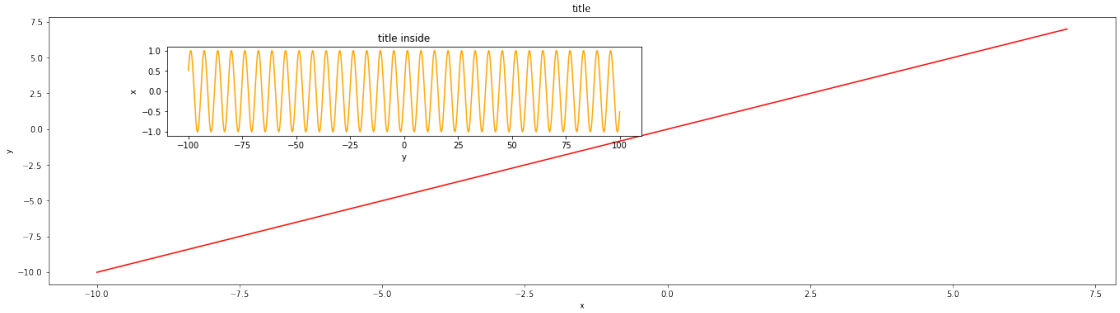
```
In [118]: ## Definimos la función que pintaremos
          eje_x = np.linspace(-100, 100, 100)
          eje_y = np.cos(eje_x)

          plt.subplot(2,2,1)
          plt.plot(eje_x, eje_y, 'r*-')
          plt.subplot(2,2,2)
          plt.plot(eje_x, eje_y, 'p--')


          # Segundo subplot
          fig = plt.figure()
          ax1 = fig.add_axes([0.1, 0.5, 1.5, 2],
```

```
                    xticklabels=[], ylim=(-1.2, 1.2))
ax2 = fig.add_axes([0.1, 0.1, 0.8, 0.4],
                    ylim=(-1, 1))

eje_x = np.linspace(0, 10)
eje_y= np.sin(eje_x)
plt.plot(eje_x,eje_y)

#ultimo subplot
fig = plt.figure()
EJEX = [-10,-9,-8,-7,-6,-5,-4,-2,0,1, 2, 3, 4, 5, 6, 7]
EJEY = [-10,-9,-8,-7,-6,-5,-4,-2,0,1, 2, 3, 4, 5, 6, 7]
axes1 = fig.add_axes([0.1, 0.1, 0.9, 0.9]) # main axes
axes2 = fig.add_axes([0.2, 0.6, 0.4, 0.3]) # inset axes
# FIGURA GRANDE
axes1.plot(EJEX, EJEY, 'r')
axes1.set_xlabel('x')
axes1.set_ylabel('y')
axes1.set_title('title')
# Interno
x = np.linspace(-100,100,4000)
y = np.sin(x)
axes2.plot(x, y, 'orange')
axes2.set_xlabel('y')
axes2.set_ylabel('x')
axes2.set_title('title inside');
```








title
title inside
x
y
x
y