Taller 6

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 27-mar-2020 11:59 PM

Santiago Ortiz Ortiz

[santiago.ortizo@urosario.edu.co (mailto:santiago.ortizo@urosario.edu.co)]

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.)

Resuelva la parte 1 de este documento (http://www.math.pitt.edu/~sussmanm/3040Summer14/exercisesII.pdf).

Solución Parte I:

In [27]:
```
# Importar paquetes
import numpy as np
import scipy.linalg as la # ¿Para qué? R// Hace lo mismo que Numpy con algunas m
import matplotlib.pyplot as plt
import pandas as pd
```

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

In [106]:
```
x = 9
```

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

In [107]:
```
np.square(x) # Este comando sirve para sacar el cuadrado de X
```

Out[107]:
```
81
```

In [114]:
```
x**(2) # De forma normal
```

Out[114]:
```
81
```

In [110]:
```
np.power(x,3) # Este comando sirve para sacar el cubo de X
```

Out[110]:
```
729
```

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

In [16]:
```
theta = 90
```

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

In [20]:
```
# Por la forma funcional del seno(theta) es claro ver que en 90 grados debería se
print("El seno de theta es:", np.sin(theta))
# Por la forma funcional del cos(theta) es claro ver que en 90 grados debería ser
print("El coseno de theta es:", np.cos(theta))
```

```
El seno de theta es: 0.8939966636005579
El coseno de theta es: -0.4480736161291701
```

In [21]:
```
# Note que nos da como resultado un número diferente a 1. Lo cual nos dice que se
# Por regla de 3, sabemos que si 180 grados es igual a PI, entonces si queremos p
print("Calculando el angulo en radianes:")
print("El seno de theta es:", np.sin(theta*(np.pi/180)))
print("El coseno de theta es:", np.cos(theta*(np.pi/180)))
```

```
Calculando el angulo en radianes:
El seno de theta es: 1.0
El coseno de theta es: 6.123233995736766e-17
```

Las funciones np.sin() y np.cos() toman los parámetros 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 [22]:
```
meshPoints = np.linspace(-1,1,500) # El primer parámetro es el inicio, el segund
```

In [25]:
```
meshPoints # Crea un vector con 500 elementos.
```

Out[25]:
```
array([-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,
```

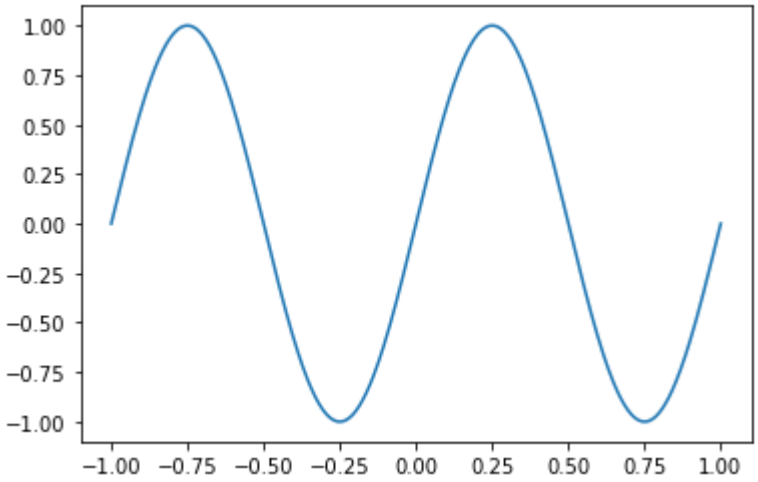
6. What expression will yield the value of the 53th element of meshPoints? What is this value?

In [29]:
```
# Como numpy crea arreglos o tipos de listas n-dimensionales, se puede utilizar 
# Como quiero el elemento 53, debo ir a la posición 52 (recuerde que es n-1)
print("El elemento 53 de la lista meshPoints (que esta ubicado en la posición 52
```

```
El elemento 53 de la lista meshPoints (que esta ubicado en la posición 52) es:,
-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 [32]:
```
plt.plot(meshPoints,np.sin(2*np.pi*meshPoints)); # Hicimos una corrección: np.pi
plt.savefig('sin_meshPoints.jpg')
```



Resuelva los ejercicios de las secciones 4.1, 5.1, 6.1, 7.4 y 8.5 de [este documento (http://www.python-academy.com/download/pycon2012/matplotlib_handout.pdf)](http://www.python-academy.com/download/pycon2012/matplotlib_handout.pdf).

4.1

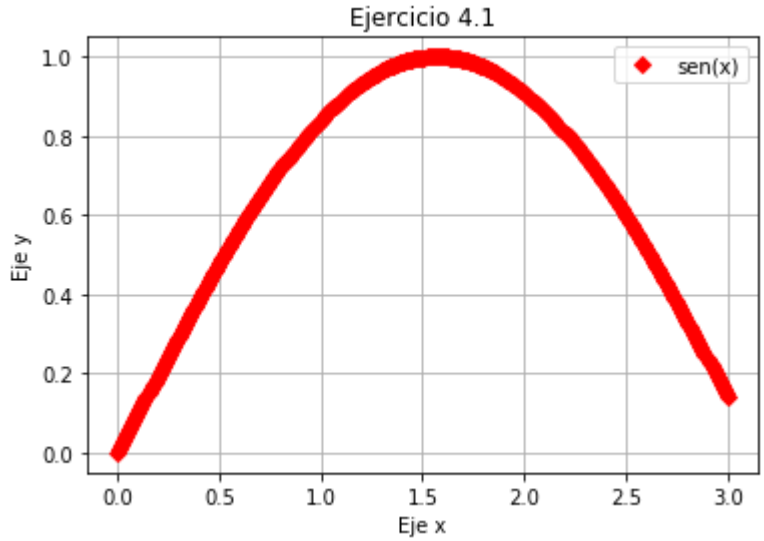
1. Plot a simple graph of a sinus function in the range 0 to 3 with a step size of 0.01.
2. Make the line red. Add diamond-shaped markers with size of 5.
3. Add a legend and a grid to the plot.

In [45]:
```
x = np.arange(0,3.01,0.01)
```

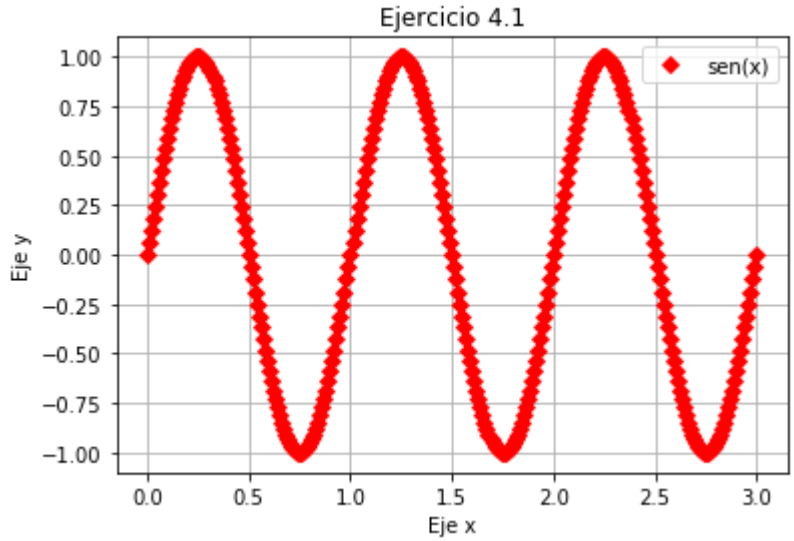
In [46]:
```
x
```

Out[46]:
```
array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99, 1.  , 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09,
       1.1 , 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19, 1.2 ,
       1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29, 1.3 , 1.31,
       1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39, 1.4 , 1.41, 1.42,
       1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49, 1.5 , 1.51, 1.52, 1.53,
       1.54, 1.55, 1.56, 1.57, 1.58, 1.59, 1.6 , 1.61, 1.62, 1.63, 1.64,
       1.65, 1.66, 1.67, 1.68, 1.69, 1.7 , 1.71, 1.72, 1.73, 1.74, 1.75,
       1.76, 1.77, 1.78, 1.79, 1.8 , 1.81, 1.82, 1.83, 1.84, 1.85, 1.86,
       1.87, 1.88, 1.89, 1.9 , 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97,
       1.98, 1.99, 2.  , 2.01, 2.02, 2.03, 2.04, 2.05, 2.06, 2.07, 2.08,
       2.09, 2.1 , 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19,
       2.2 , 2.21, 2.22, 2.23, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29, 2.3 ,
       2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37, 2.38, 2.39, 2.4 , 2.41,
       2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49, 2.5 , 2.51, 2.52,
       2.53, 2.54, 2.55, 2.56, 2.57, 2.58, 2.59, 2.6 , 2.61, 2.62, 2.63,
       2.64, 2.65, 2.66, 2.67, 2.68, 2.69, 2.7 , 2.71, 2.72, 2.73, 2.74,
       2.75, 2.76, 2.77, 2.78, 2.79, 2.8 , 2.81, 2.82, 2.83, 2.84, 2.85,
       2.86, 2.87, 2.88, 2.89, 2.9 , 2.91, 2.92, 2.93, 2.94, 2.95, 2.96,
       2.97, 2.98, 2.99, 3.  ])
```

In [55]:
```
plt.title("Ejercicio 4.1")
plt.xlabel("Eje x")
plt.ylabel("Eje y")
plt.plot(x, np.sin(x),"rD", markersize=5, label = "sen(x)")
plt.grid(True)
plt.legend();
```



In [54]:
```
plt.title("Ejercicio 4.1")
plt.xlabel("Eje x")
plt.ylabel("Eje y")
plt.plot(x, np.sin(x*2*np.pi),"rD", markersize=5, label = "sen(x)")
plt.grid(True)
plt.legend();
```



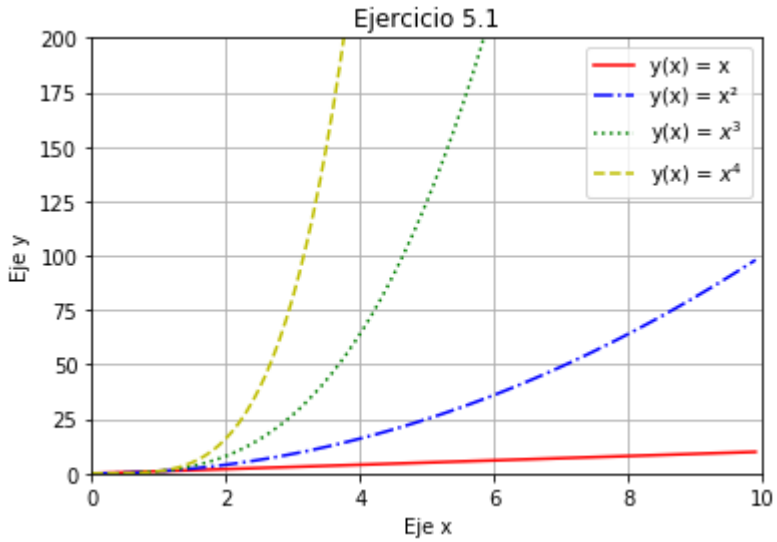
5.1

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 [68]:
```
# Crear los números
x_values = np.arange(0,10, 0.1)
```

In [69]:
```
# Funciones
y_x1 = x_values
y_x2 = x_values**2
y_x3 = x_values**3
y_x4 = x_values**4
```

In [75]:
```
# Graficar
plt.title("Ejercicio 5.1")
plt.xlabel("Eje x")
plt.ylabel("Eje y")
plt.xlim(0,10) #Rango de ejes
plt.ylim(0,200)
plt.plot(x_values, y_x1,"r-", markersize=2, label = "y(x) = x")
plt.plot(x_values, y_x2,"b-.", markersize=3, label = f"y(x) = x\N{SUPERSCRIPT TW
plt.plot(x_values, y_x3,"g:", markersize=5, label = f"y(x) = $x^3$")
plt.plot(x_values, y_x4,"y--", markersize=1, label = f"y(x) = $x^4$") # Como en
plt.grid(True)
plt.legend();
```

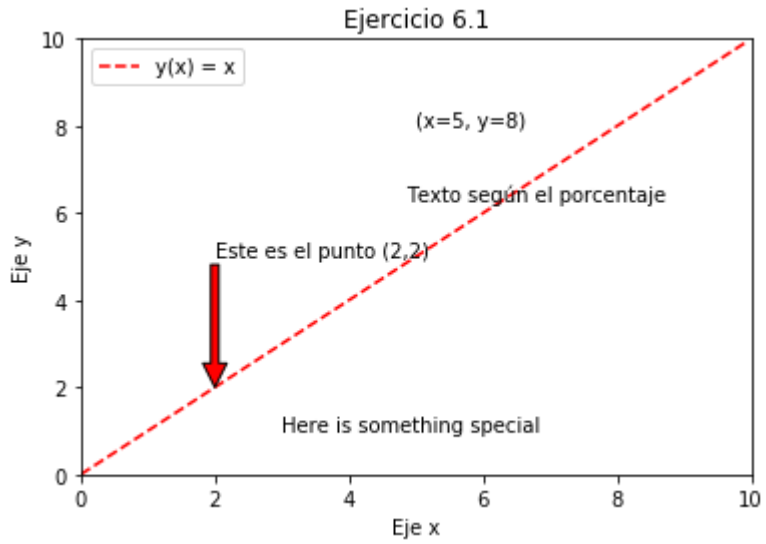


6.1

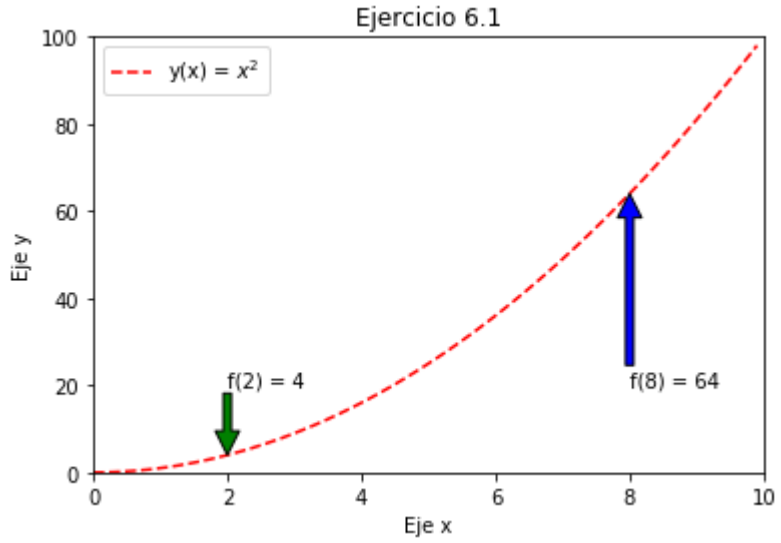
1. Annotate a line at two places with text. Use green and red arrows and align it according to figure points and data.

PRUEBA

In [101]:
```
plt.title("Ejercicio 6.1")
plt.xlabel("Eje x")
plt.ylabel("Eje y")
plt.xlim(0,10) #Rango de ejes
plt.ylim(0,10)
plt.plot(x_values, y_x1,"r--", markersize=2, label = "y(x) = x")
plt.text(5, 8, '(x=5, y=8)') # Para poner algún texto en la posición x,y
plt.figtext(0.5, 0.6, 'Texto según el porcentaje') # Toma la imagen completa com
plt.annotate('Here is something special', xy = (3, 1)) # Funciona igual que text
plt.annotate("Este es el punto (2,2)", xy = (2, 2), xytext=(2,5),arrowprops={'fa
plt.legend();
```



In [108]:
```
plt.title("Ejercicio 6.1")
plt.xlabel("Eje x")
plt.ylabel("Eje y")
plt.xlim(0,10) #Rango de ejes
plt.ylim(0,100)
plt.plot(x_values, y_x2,"r--", markersize=2, label = "y(x) = $x^2$")
plt.annotate('f(2) = 4', xy=(2,4), xytext=(2, 20),arrowprops=dict(facecolor='gre
plt.annotate('f(8) = 64', xy=(8,64), xytext=(8, 20),arrowprops=dict(facecolor='b
plt.legend();
```



7.4

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 [8]:
```
import datetime
import pandas as pd
import datetime
from random import randint
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
```

In [9]:
```
# Utilizamos la función datetime.datetime que funciona así:
# datetime(year, month, day[, hour[, minute[, second[, microsecond[,tzinfo]]]]]),
primer_dia = datetime.datetime(2017,1,1) #función datetime, método .datetime
```

In [10]:
```
primer_dia # Año 2017, Mes enero, día primero. No pusimos horas.
```

Out[10]:
```
datetime.datetime(2017, 1, 1, 0, 0)
```

In [13]:
```
# Número de días que va a tener la serie
numero_dias = 365
dias = []
for i in range (numero_dias + 1):
    dia_siguiente = primer_dia + datetime.timedelta(days = i) #Toma el día inici
                                                              # se utiliza la fu
    print(dia_siguiente)
    dia_siguiente = dia_siguiente.strftime("%Y-%m-%d") # Formato en que aparecer
    print(dia_siguiente)
    dias.append(dia_siguiente)
```

```
2017-03-17
2017-03-18 00:00:00
2017-03-18
2017-03-19 00:00:00
2017-03-19
2017-03-20 00:00:00
2017-03-20
2017-03-21 00:00:00
2017-03-21
2017-03-22 00:00:00
2017-03-22
2017-03-23 00:00:00
2017-03-23
2017-03-24 00:00:00
2017-03-24
2017-03-25 00:00:00
2017-03-25
2017-03-26 00:00:00
2017-03-26
2017-03-27 00:00:00
```

In [14]:
```
dias
```

```
 '2017-07-24',
 '2017-07-25',
 '2017-07-26',
 '2017-07-27',
 '2017-07-28',
 '2017-07-29',
 '2017-07-30',
 '2017-07-31',
 '2017-08-01',
 '2017-08-02',
 '2017-08-03',
 '2017-08-04',
 '2017-08-05',
 '2017-08-06',
 '2017-08-07',
 '2017-08-08',
 '2017-08-09',
 '2017-08-10',
 '2017-08-11',
 '2017-08-12'.
```

In [17]:
```
import random
```

In [23]:
```
values = []
for i in range(len(dias)):
    values.append(random.randint(0,100))
```

In [45]:
```
datos = pd.DataFrame() # Creamos un dataframe
datos['valor'] = values # Creamos la columna "valor"
datos = datos.set_index(pd.to_datetime(dias)) #set_index crea filas utilizando u
```

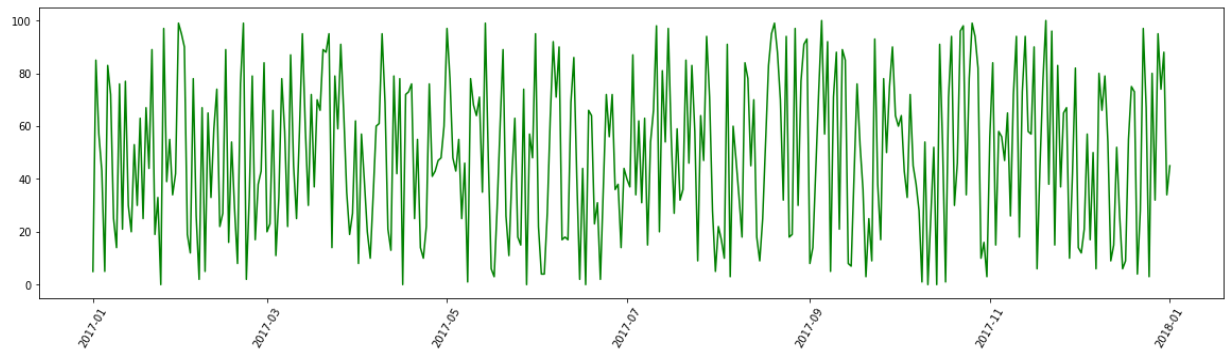
In [49]:
```
datos
```

Out[49]:

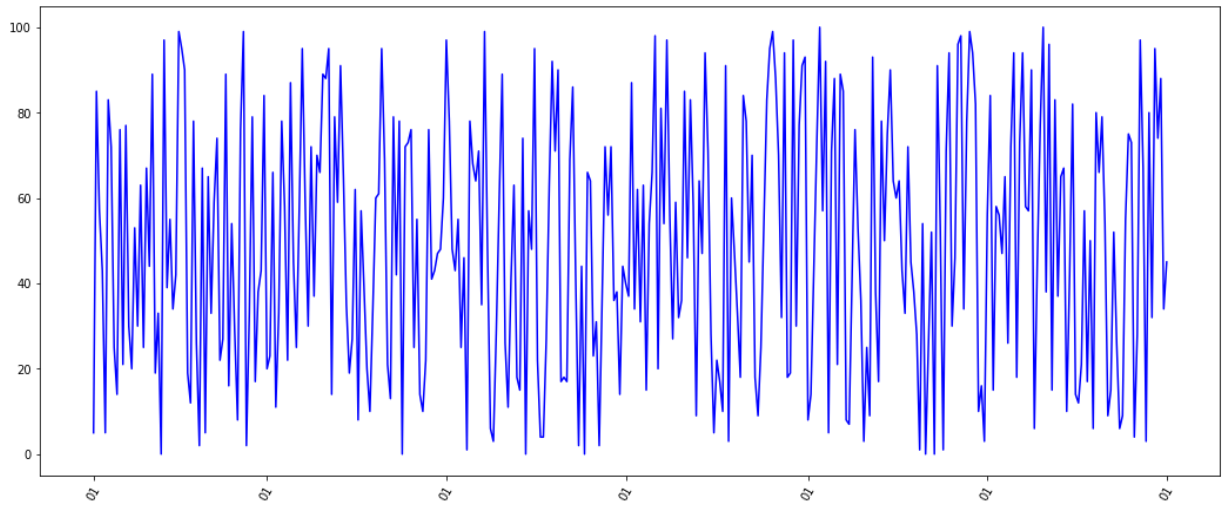
	valor
2017-01-01	5
2017-01-02	85
2017-01-03	57
2017-01-04	43
2017-01-05	5
...	...
2017-12-28	95
2017-12-29	74
2017-12-30	88
2017-12-31	34
2018-01-01	45

366 rows × 1 columns

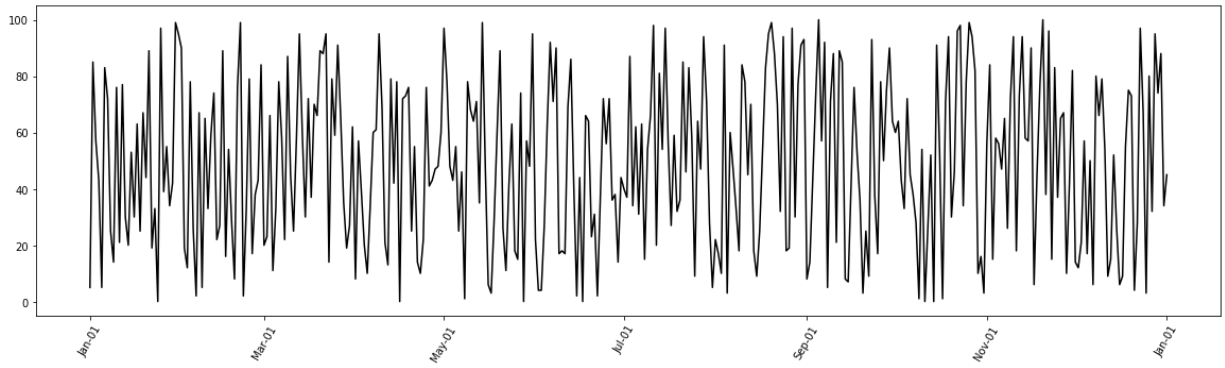
In [54]:
```
plt.xticks(rotation=60) # ángulo de 60 grados
plt.plot(datos,"green")
plt.rcParams["figure.figsize"] = [18.0, 7.0]
```



In [58]:
```
fig, ax = plt.subplots()
fig.subplots_adjust(bottom=0.09)
plt.xticks(rotation=60)
plt.plot(datos,"blue")
plt.rcParams["figure.figsize"] = [18.0, 7.0]
ax.xaxis.set_major_formatter(DateFormatter('%d'))
```



In [61]:
```
fig, ax = plt.subplots()
fig.subplots_adjust(bottom=0.07)
plt.xticks(rotation=60)
plt.plot(datos,"black")
plt.rcParams["figure.figsize"] = [20.0, 5.0]
ax.xaxis.set_major_formatter(DateFormatter('%b-%d')) # Cambiamos el número de mes
```

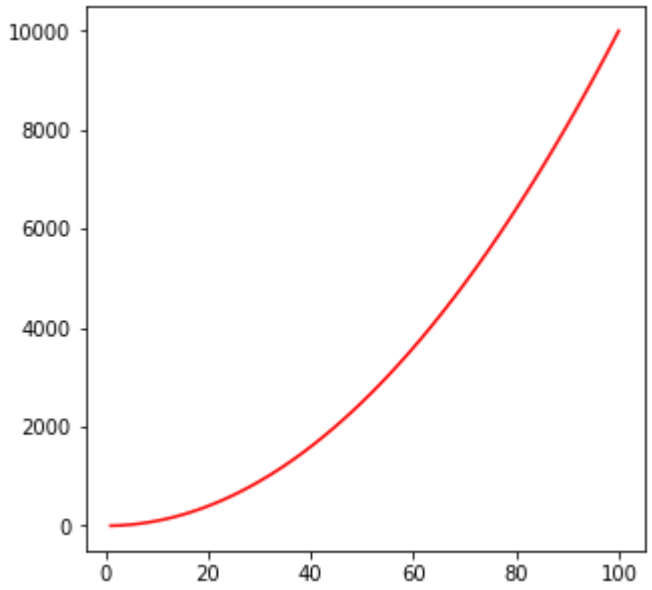


8.5

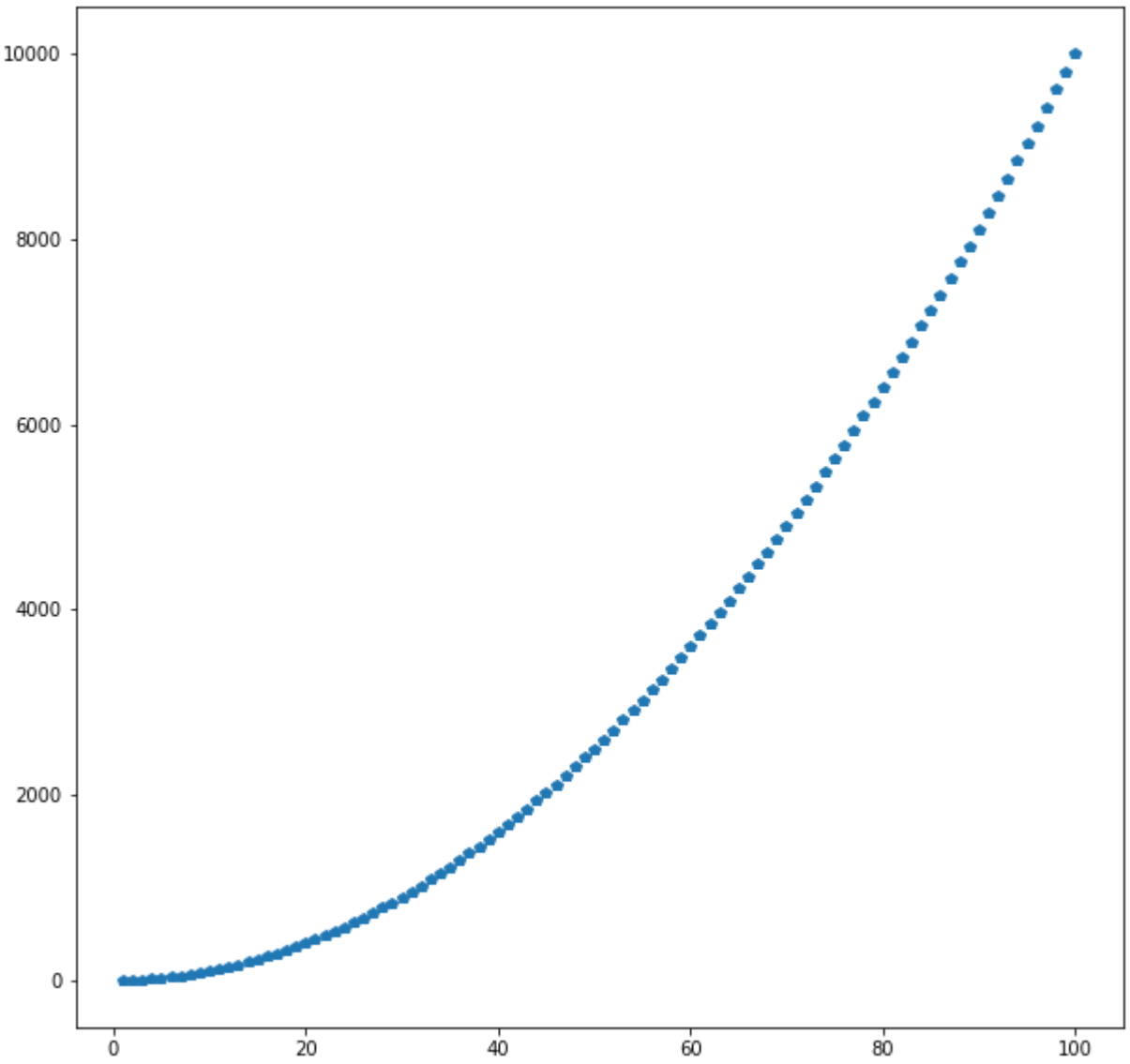
1. Draw two figures, one 5 by 5, one 10 by 10 inches.
2. Add four subplots to one figure. Add labels and ticks only to the outermost axes.
3. Place a small plot in one bigger plot.

In [74]:
```
# Haremos el mismo plot con tamaños diferentes
x_valores = np.arange(1,101)
y_valores = x_valores**2
```

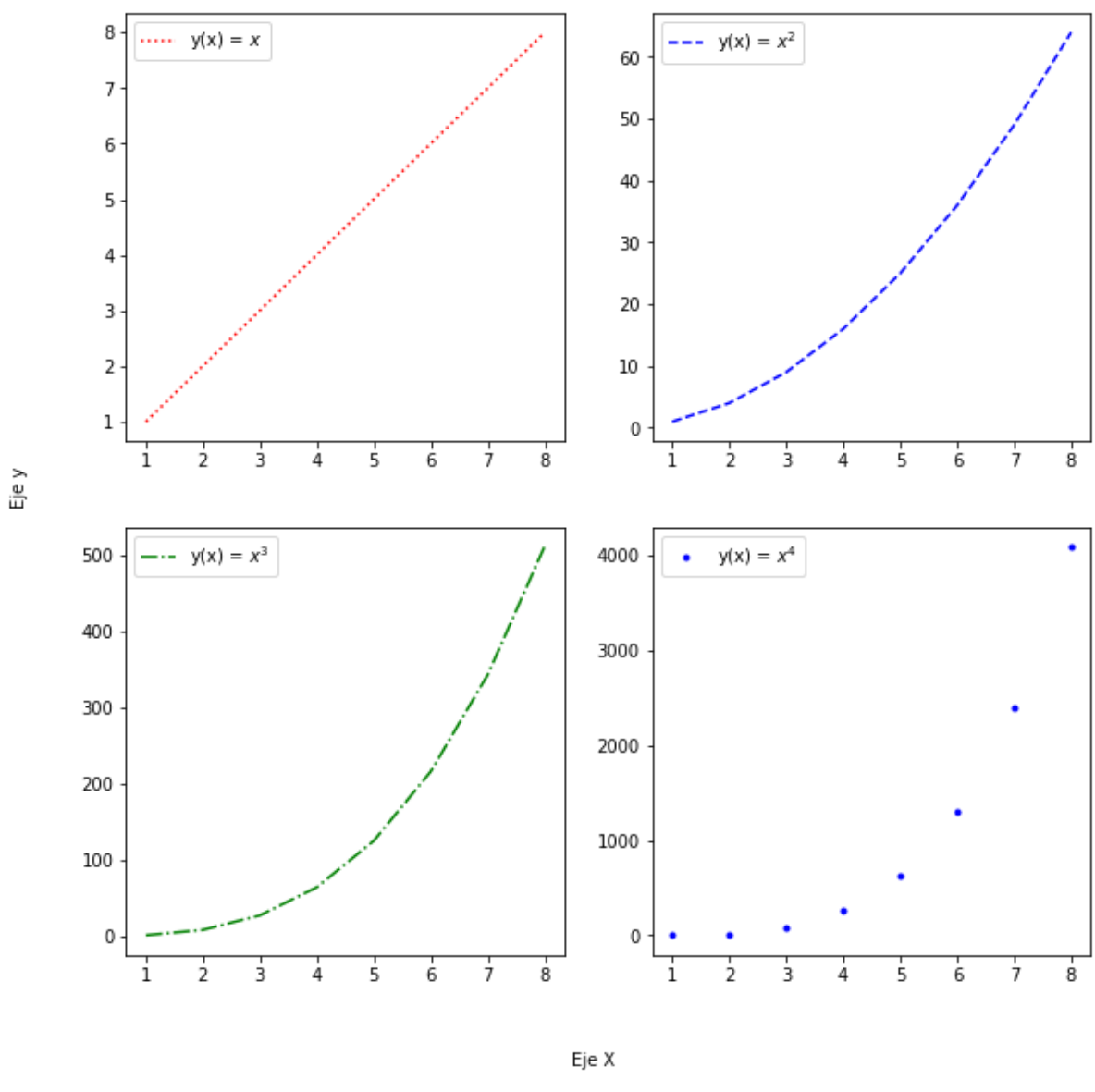
In [75]:
```
plt.rcParams["figure.figsize"] = [5.0, 5.0]
plt.plot(x_valores, y_valores, 'r');
```



In [76]:
```
plt.rcParams["figure.figsize"] = [10.0, 10.0]
plt.plot(x_valores, y_valores, 'p');
```

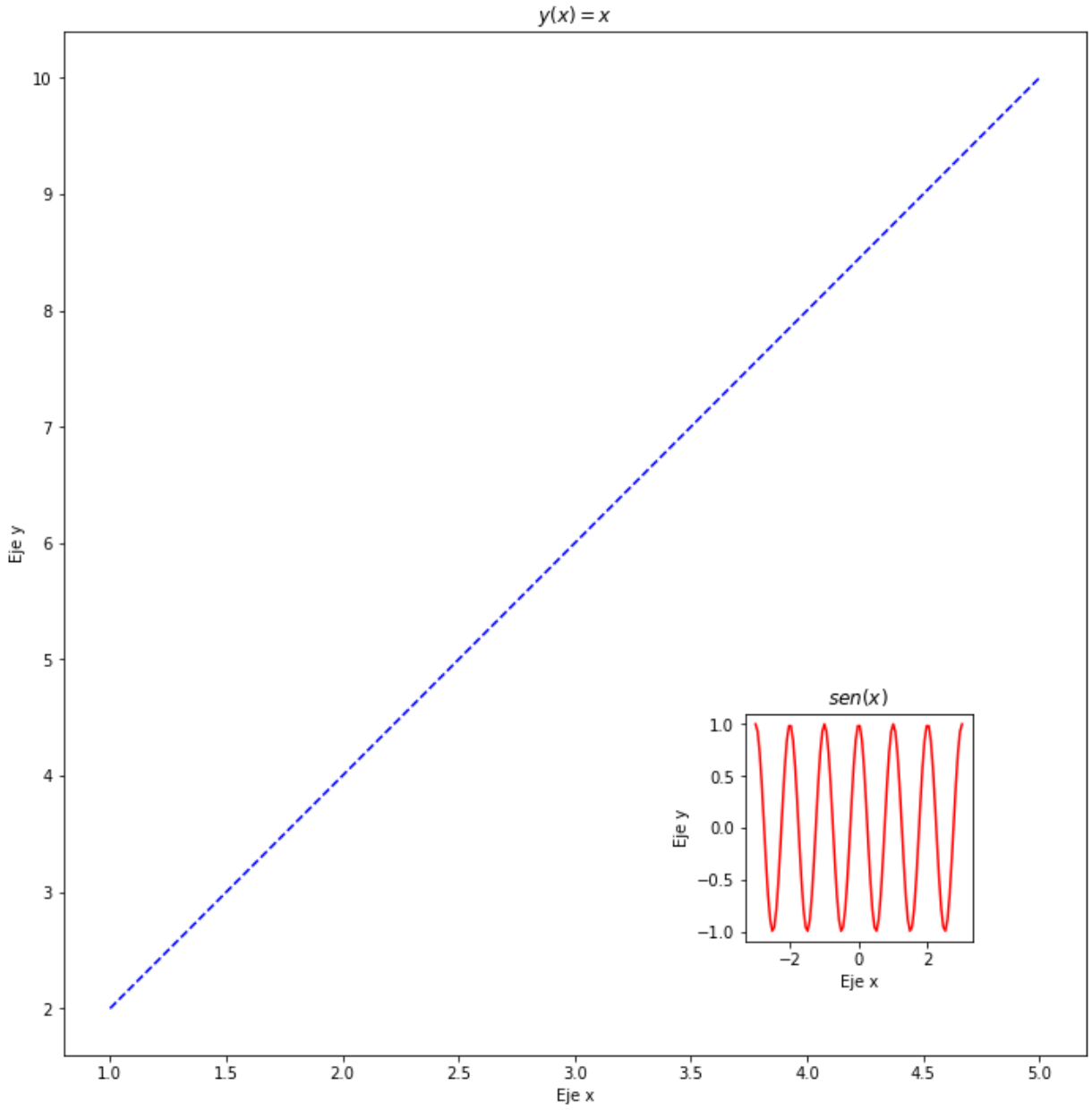


In [91]:
```
2.
# Funciones
x_values = np.arange(1,9)
y_x1 = x_values
y_x2 = x_values**2
y_x3 = x_values**3
y_x4 = x_values**4

fig, ax = plt.subplots(2, 2,
                       gridspec_kw={
                           'width_ratios': [5,5],
                           'height_ratios': [5,5]})

ax[0][0].plot(x_values, y_x1, 'r:',label = f"y(x) = $x$")
ax[0][1].plot(x_values, y_x2, 'b--', label = f"y(x) = $x^2$")
ax[1][0].plot(x_values, y_x3, 'g-.', label = f"y(x) = $x^3$")
ax[1][1].plot(x_values, y_x4, 'b.',label = f"y(x) = $x^4$")
fig.text(0.5, 0.04, 'Eje X', va='center', ha='center')
fig.text(0.04, 0.5, 'Eje y', va='center',rotation='vertical', ha='center')
ax[0][0].legend([f"y(x) = $x$"])
ax[0][1].legend([f"y(x) = $x^2$"])
ax[1][0].legend([f"y(x) = $x^3$"])
ax[1][1].legend([f"y(x) = $x^4$"])
plt.show()
```



In [93]:
```
#3
eje_x = np.arange(1,6)
eje_y = np.arange(1,6)*2
```

In [111]:
```
fig = plt.figure()
axes1 = fig.add_axes([0.1, 0.1, 0.9, 0.9]) # Figura principal
axes2 = fig.add_axes([0.7, 0.2, 0.2, 0.2]) # La que va adentro # (Mover en x, mov
# Fig. principal
axes1.plot(eje_x, eje_y, 'b--')
axes1.set_xlabel('Eje x') # poner nombre a ejes
axes1.set_ylabel('Eje y')
axes1.set_title(f"$y(x) = x$")

# Fig. interna
x = np.linspace(-3,3,100)
y = np.cos(x*2*np.pi)
axes2.plot(x, y, 'red')
axes2.set_xlabel('Eje x')
axes2.set_ylabel('Eje y')
axes2.set_title('$sen(x)$');
```



In [ ]: