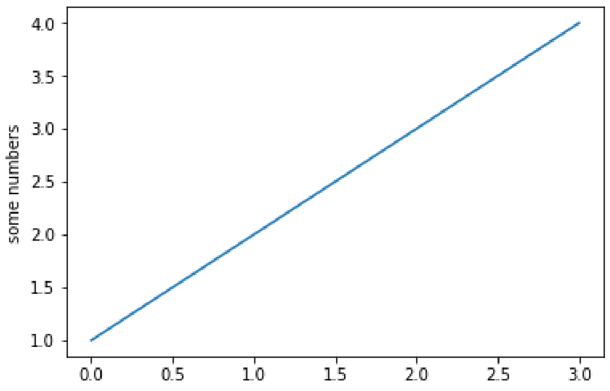
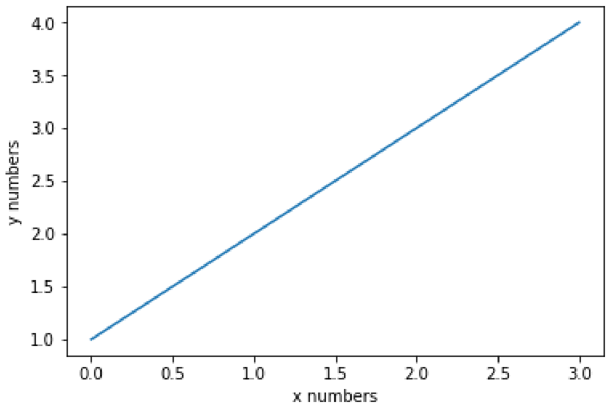
In [3]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



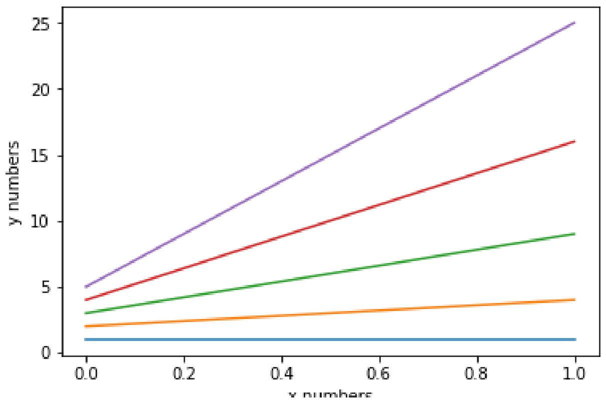
In [4]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.xlabel('x numbers')
plt.ylabel('y numbers')
plt.show()
```



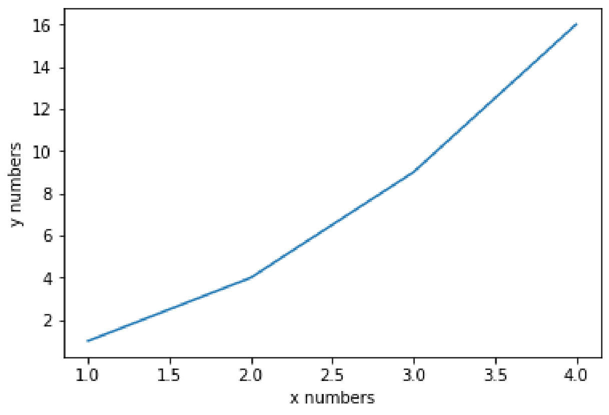
In [7]:

```
import matplotlib.pyplot as plt
plt.plot([[1,2,3,4,5],[1,4,9,16,25]])
plt.xlabel('x numbers')
plt.ylabel('y numbers')
plt.show()
```



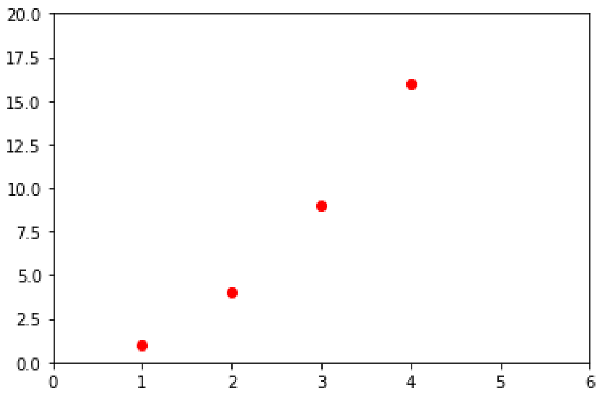
In [8]:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.xlabel('x numbers')
plt.ylabel('y numbers')
plt.show()
```



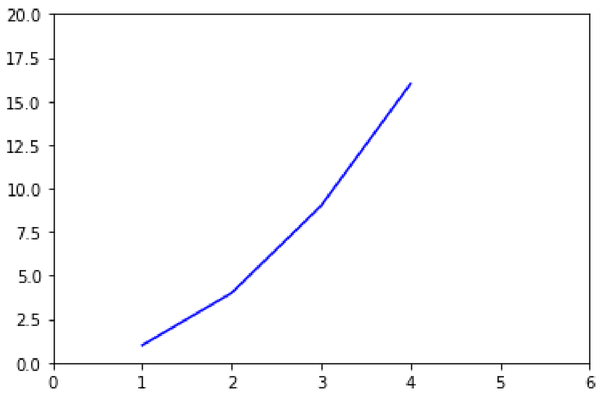
In [9]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



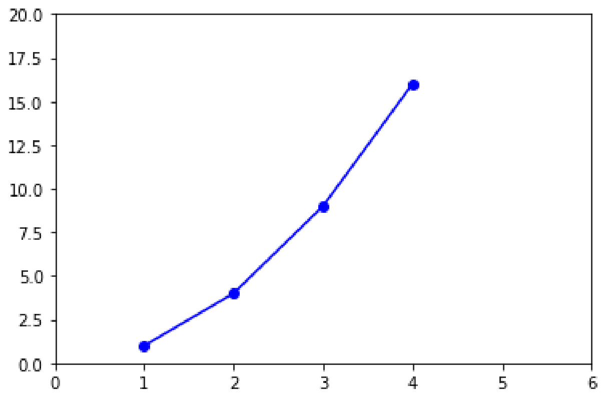
In [10]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'b-')
plt.axis([0, 6, 0, 20])
plt.show()
```



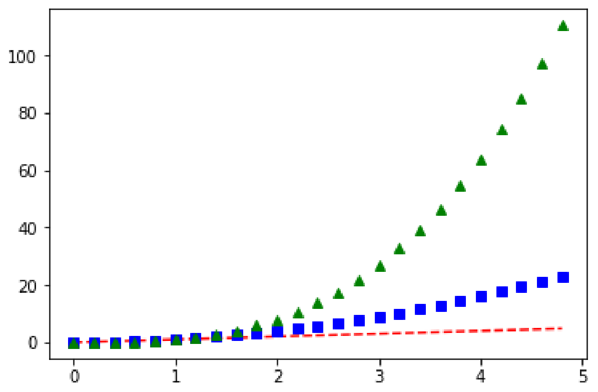
In [13]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'bo-')
plt.axis([0, 6, 0, 20])
plt.show()
```



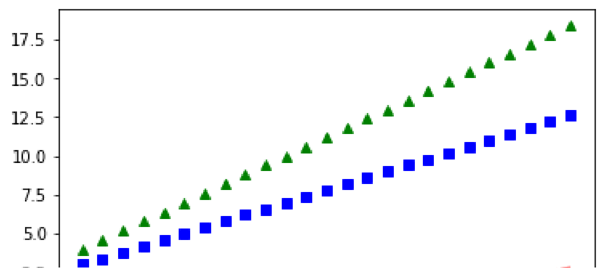
In [14]:

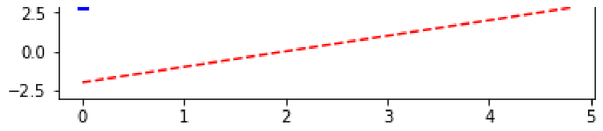
```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



In [17]:

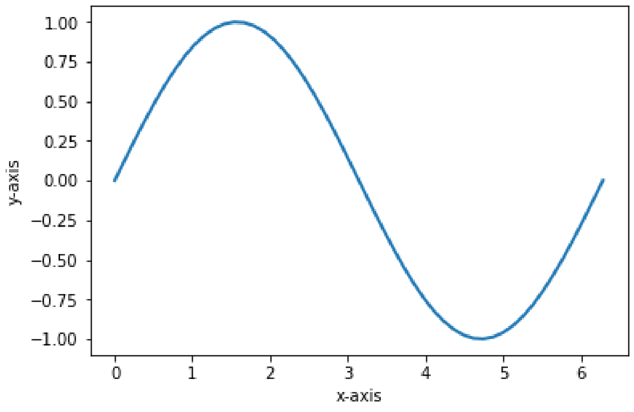
```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t-2, 'r--', t, 2*t+3, 'bs', t, 3*t+4, 'g^')
plt.show()
```





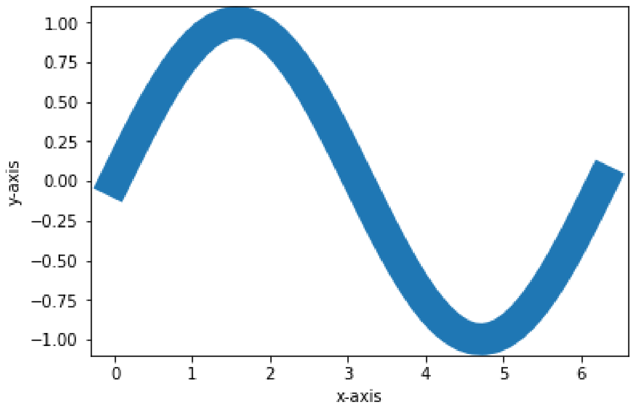
In [30]:

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

# make some data
x = np.linspace(0, 2*np.pi)

y = np.sin(x)
plt.plot(x, y, linewidth=2.0)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```



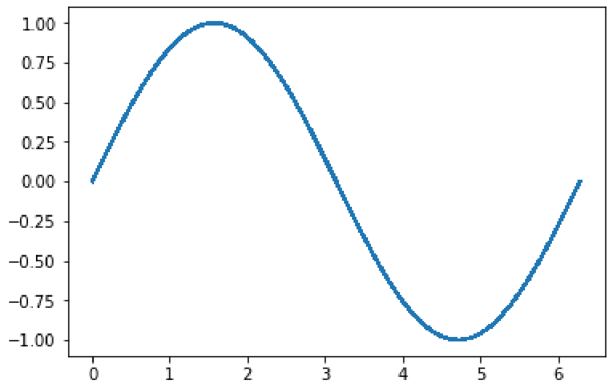
In [31]:

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

# make some data
x = np.linspace(0, 2*np.pi)

y = np.sin(x)
plt.plot(x, y, linewidth=20.0)
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```



In [34]:

```
line, = plt.plot(x, y, '-')
line.set_antialiased(False)
```

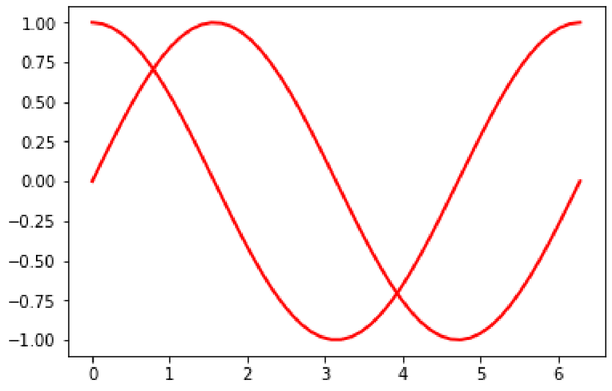


In [38]:

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

# make some data
x = np.linspace(0, 2*np.pi)

y1 = np.sin(x)
y2= np.cos(x)
lines = plt.plot(x, y1, x, y2)
# use keyword args
plt.setp(lines, color='r', linewidth=2.0)
# or MATLAB style string value pairs
plt.setp(lines, 'color', 'r', 'linewidth', 2.0)
```

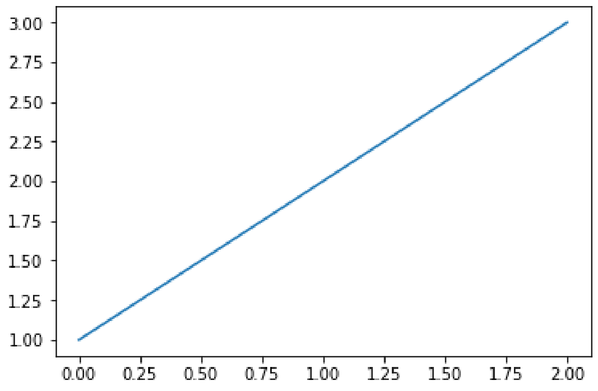
Out[38]:

```
[None, None, None, None]
```



In [39]:

```
lines = plt.plot([1, 2, 3])
```



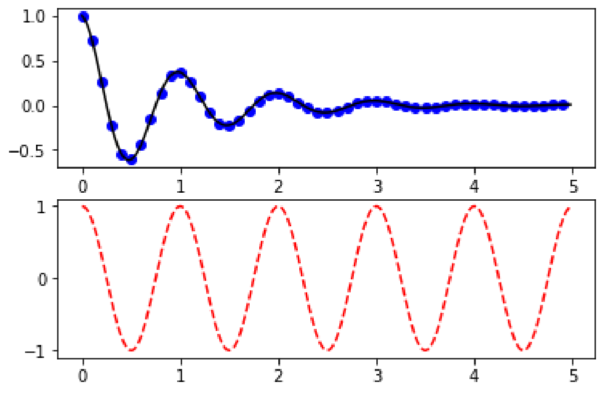
In [40]:

```
plt.setp(lines)
  alpha: float
  animated: [True | False]
  antialiased or aa: [True | False]
  ...snip
```

```
  File "<ipython-input-40-8d83d34aba2a>", line 2
    alpha: float
    ^
IndentationError: unexpected indent
```

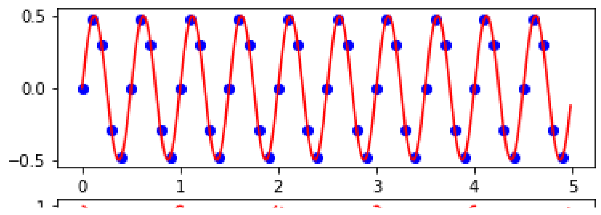
In [41]:

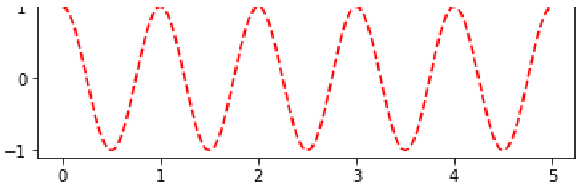
```
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



In [44]:

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

def f(t):
    return np.sin(2*np.pi*t)*np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'r')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```





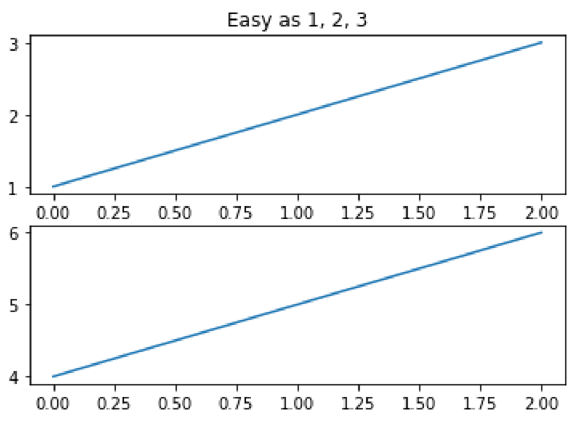
In [45]:

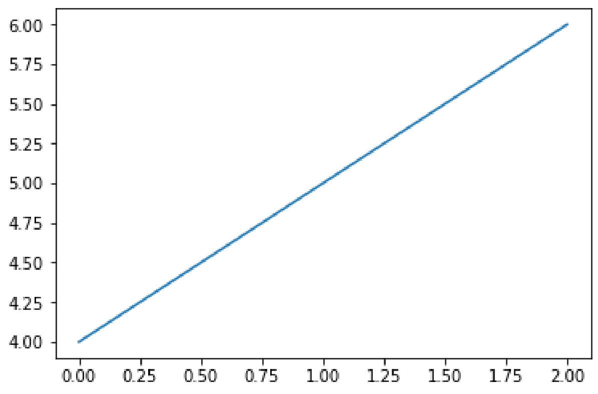
```
import matplotlib.pyplot as plt
plt.figure(1)                # the first figure
plt.subplot(211)             # the first subplot in the first figure
plt.plot([1, 2, 3])
plt.subplot(212)             # the second subplot in the first figure
plt.plot([4, 5, 6])


plt.figure(2)                # a second figure
plt.plot([4, 5, 6])          # creates a subplot(111) by default

plt.figure(1)                # figure 1 current; subplot(212) still current
plt.subplot(211)             # make subplot(211) in figure1 current
plt.title('Easy as 1, 2, 3') # subplot 211 title
```

```
C:\Users\Acer\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107:
MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes currently
reuses the earlier instance.  In a future version, a new instance will always be created and retur
ned.  Meanwhile, this warning can be suppressed, and the future behavior ensured, by passing a uni
que label to each axes instance.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

Out[45]:

```
Text(0.5,1,'Easy as 1, 2, 3')
```



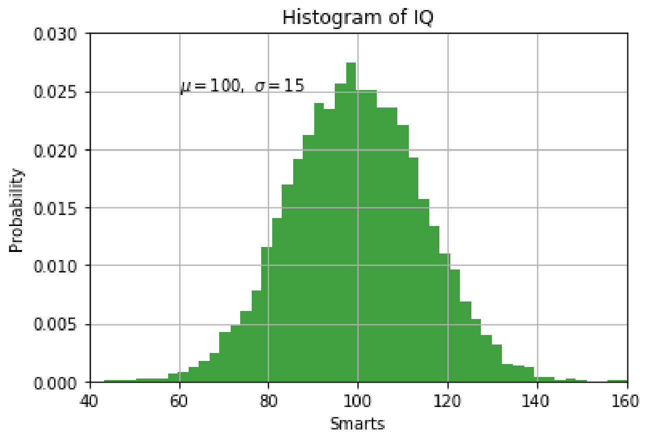


In [46]:

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

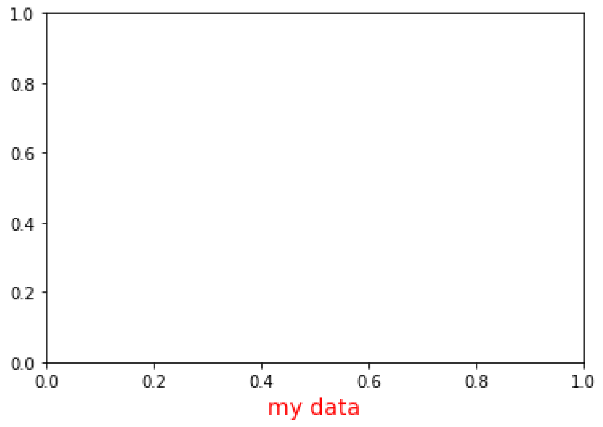
```
# Fixing random state for reproducibility
np.random.seed(19680801)

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

```
C:\Users\Acer\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6571: UserWarning: The 'normed' 
kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [47]:

```
t = plt.xlabel('my data', fontsize=14, color='red')
```

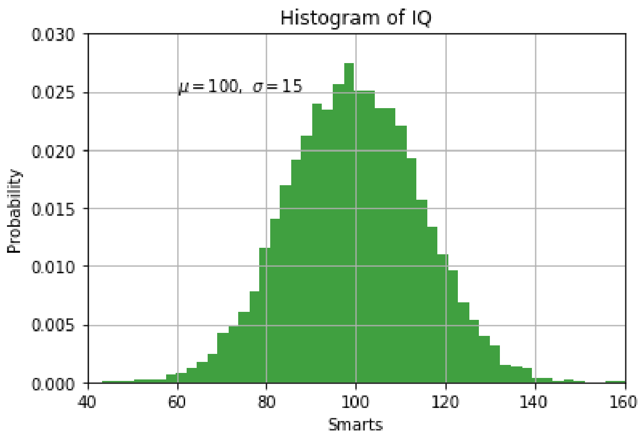


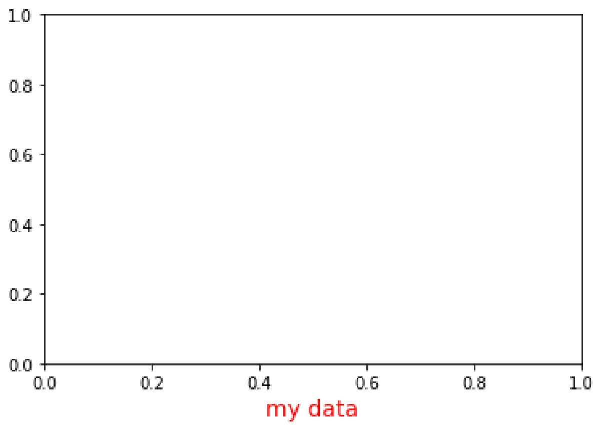
In [48]:

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

# Fixing random state for reproducibility
np.random.seed(19680801)

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)
```

```
plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
t = plt.xlabel('my data', fontsize=14, color='red')
```

```
C:\Users\Acer\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6571: UserWarning: The 'normed'
kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



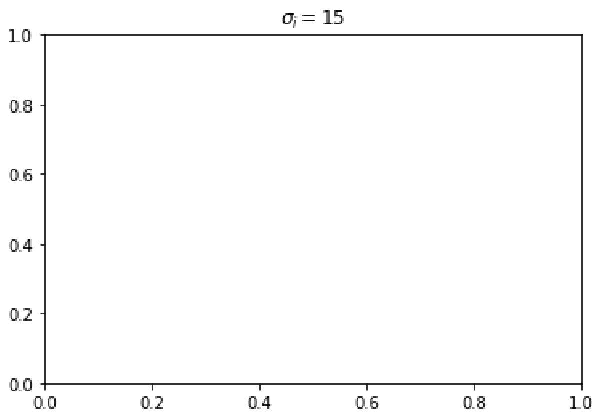


In [49]:

```
plt.title(r'$\sigma_i=15$')
```

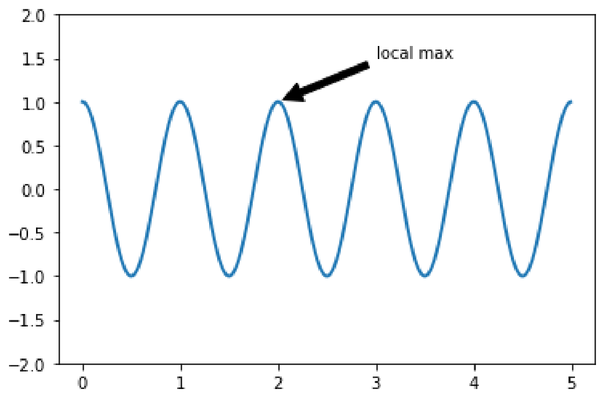
Out[49]:

```
Text(0.5,1,'$\\sigma_i=15$')
```



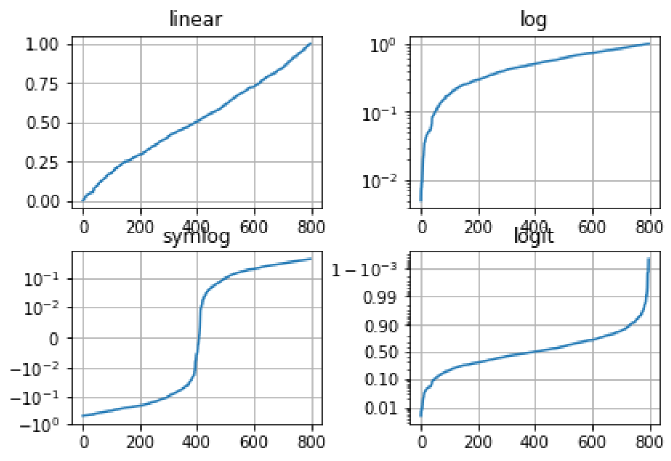
In [50]:

```python
import numpy as np
import matplotlib.pyplot as plt

ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
            arrowprops=dict(facecolor='black', shrink=0.05),
            )

plt.ylim(-2,2)
plt.show()
```



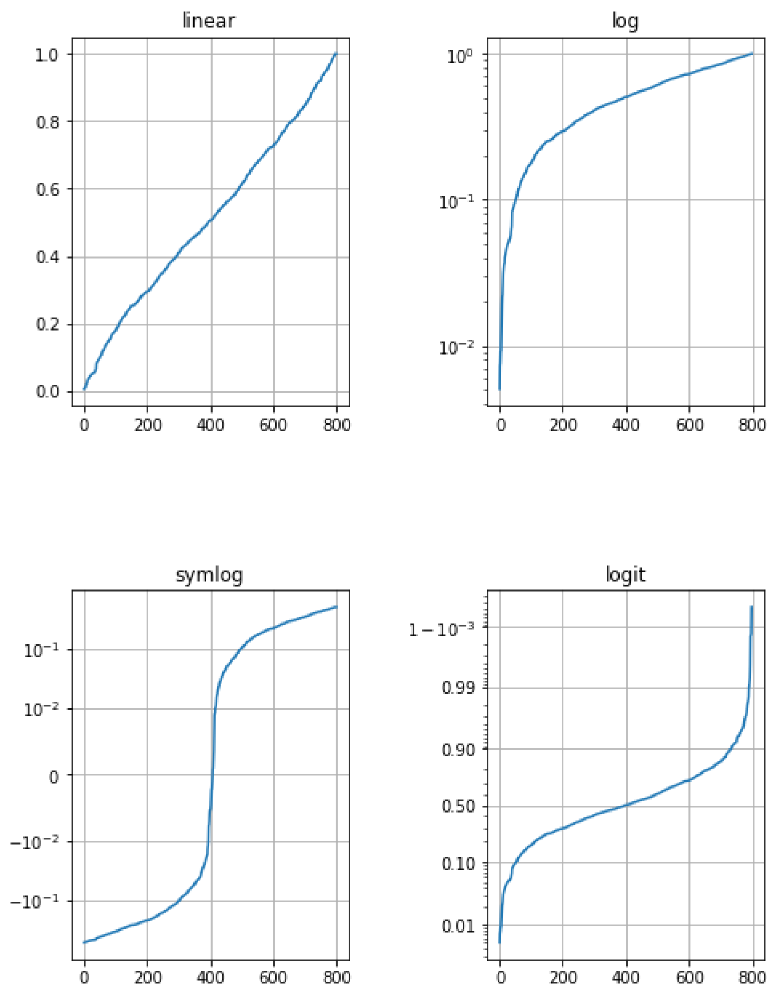
In [51]:

```python
import numpy as np
import matplotlib.pyplot as plt

from matplotlib.ticker import NullFormatter  # useful for `logit` scale

# Fixing random state for reproducibility
np.random.seed(19680801)

# make up some data in the interval ]0, 1[
y = np.random.normal(loc=0.5, scale=0.4, size=1000)
y = y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))

# plot with various axes scales
plt.figure(1)

# linear
plt.subplot(221)
plt.plot(x, y)
plt.yscale('linear')
plt.title('linear')
plt.grid(True)


# log
plt.subplot(222)
plt.plot(x, y)
plt.yscale('log')
plt.title('log')
plt.grid(True)


# symmetric log
plt.subplot(223)
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthreshy=0.01)
plt.title('symlog')
plt.grid(True)

# logit
plt.subplot(224)
plt.plot(x, y)
plt.yscale('logit')
plt.title('logit')
plt.grid(True)
# Format the minor tick labels of the y-axis into empty strings with
# `NullFormatter`, to avoid cumbering the axis with too many labels.
plt.gca().yaxis.set_minor_formatter(NullFormatter())
# Adjust the subplot layout, because the logit one may take more space
# than usual, due to y-tick labels like "1 - 10^{-3}"
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hspace=0.25,
                    wspace=0.35)

plt.show()
```



In [54]:

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

from matplotlib.ticker import NullFormatter  # useful for `logit` scale

# Fixing random state for reproducibility
np.random.seed(19680801)

# make up some data in the interval ]0, 1[
y = np.random.normal(loc=0.5, scale=0.4, size=1000)
y = y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))

# plot with various axes scales
plt.figure(1)

# linear
plt.subplot(221)
plt.plot(x, y)
plt.yscale('linear')
plt.title('linear')
plt.grid(True)


# log
plt.subplot(222)
plt.plot(x, y)
plt.yscale('log')
plt.title('log')
plt.grid(True)


# symmetric log
plt.subplot(223)
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthreshy=0.01)
plt.title('symlog')
plt.grid(True)

# logit
plt.subplot(224)
plt.plot(x, y)
plt.yscale('logit')
plt.title('logit')
plt.grid(True)
# Format the minor tick labels of the y-axis into empty strings with
# `NullFormatter`, to avoid cumbering the axis with too many labels.
plt.gca().yaxis.set_minor_formatter(NullFormatter())
# Adjust the subplot layout, because the logit one may take more space
# than usual, due to y-tick labels like "1 - 10^{-3}"
plt.subplots_adjust(top=2.5, bottom=0.5, left=0.5, right=1.5, hspace=0.5,
                    wspace=0.5)

plt.show()
```



```
In [ ]:
```