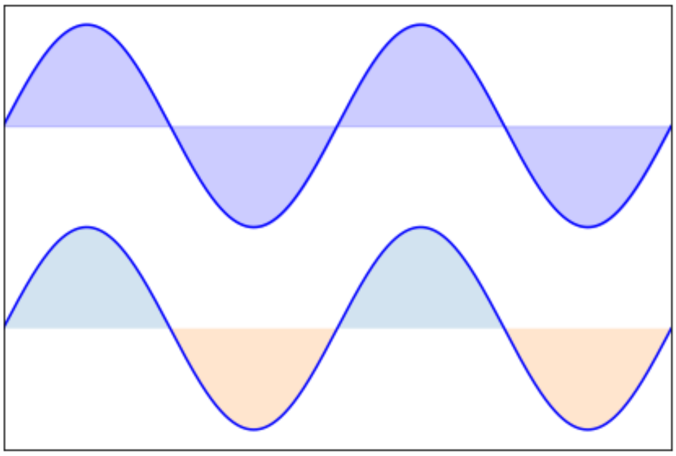
Plotting

In [34]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

1. Create the plot shown in the figure below, some helper code is provided below so you only have to fill in the missing code that creates the colored regions.

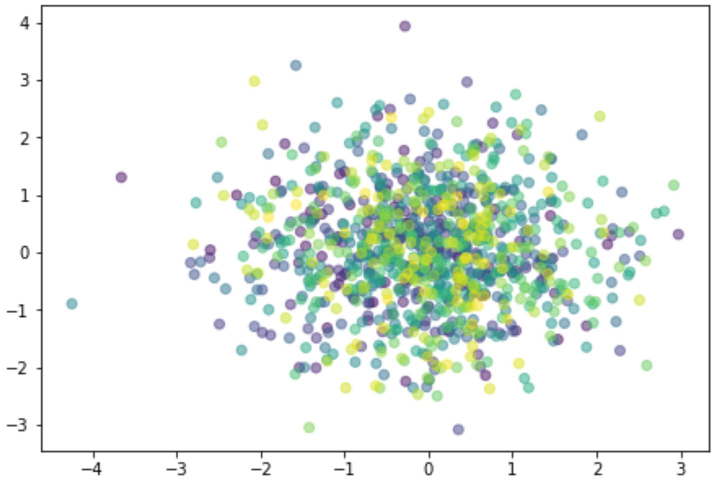
In [35]:
```
n = 1024
X = np.linspace(-np.pi,np.pi,n,endpoint=True)
Y = np.sin(2*X)

plt.axes([0.025,0.025,0.95,0.95])

plt.plot (X, Y+1, color='blue', alpha=1.00)
plt.plot (X, Y-1, color='blue', alpha=1.00)
plt.fill_between(X,Y+1,1,color="b", alpha=0.2)
plt.fill_between(X,Y-1,-1,Y>0, alpha=0.2)
plt.fill_between(X,Y-1,-1,Y<0,"r", alpha=0.2,)
plt.xlim(X[0],X[-1])
plt.xticks([])
plt.yticks([])

plt.show()
```



In [36]:
```
X = np.linspace
```

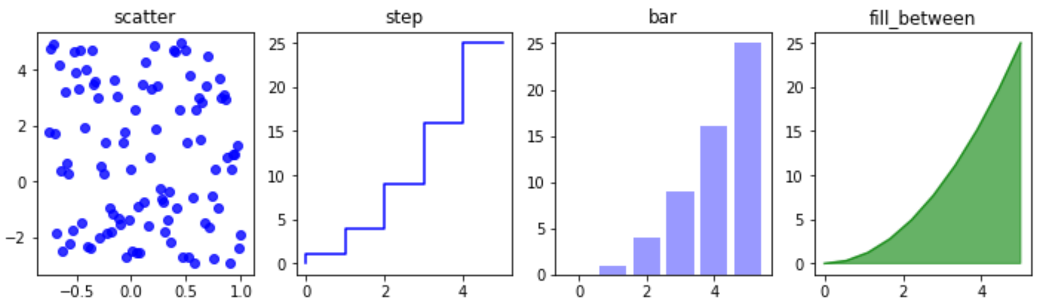
1. Recreate the plot shown below, fill in the neccessary code to add the colors and the transparency factors.

In [37]:
```
'''the code given below creates a random set of points on the graph,
fill in the appropriate arguments to recreate the given image'''

n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)

plt.axes([0.025,0.025,0.95,0.95])
plt.scatter(X,Y,c=np.arange(0,n),alpha=0.5)

plt.show()
```



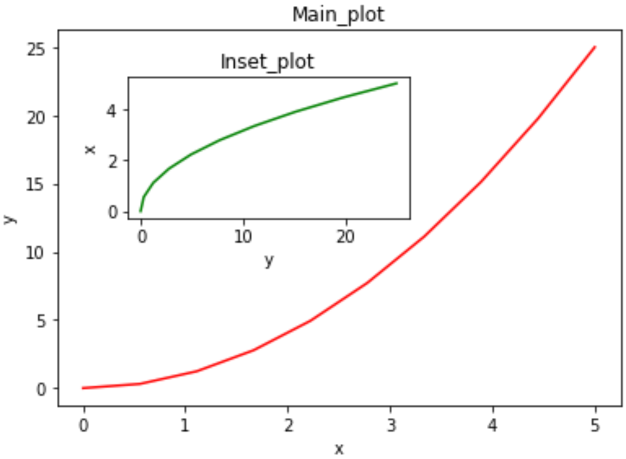
1. Recreate the different 2-D plots shown in the figure below, the axes take the values for x and x^2

In [38]:
```
import random
# use the 'n' dataset to create the step and bar graphs
n = np.array([0,1,2,3,4,5])
sqr = n**2
# the scatter plot can be obtained using the 'xx' datapoints
xx = np.linspace(-0.75, 1., 100)
yy = [random.uniform(-3,5) for x in xx]

# use the 'x' dataset to create the fill_between plot
x = np.linspace(0, 5, 10)
y = x**2

# Insert code here to add the figure and subplots
fig, axes = plt.subplots(1, 4,figsize=(12,3))

#Insert code here

axes[0].set_title("scatter")
axes[0].scatter(xx,yy,c="b",alpha=0.8)


#Insert code here

axes[1].set_title("step")
axes[1].step(n,sqr,c="b")

#Insert code here

axes[2].set_title("bar")
axes[2].bar(n,sqr,color="b",alpha=0.4)

#Insert code here

axes[3].set_title("fill_between");
axes[3].plot(x,y,c='g',alpha=0.6)
axes[3].fill(np.append(x,5),np.append(y,0),c='g',alpha=0.6)

plt.show()
```



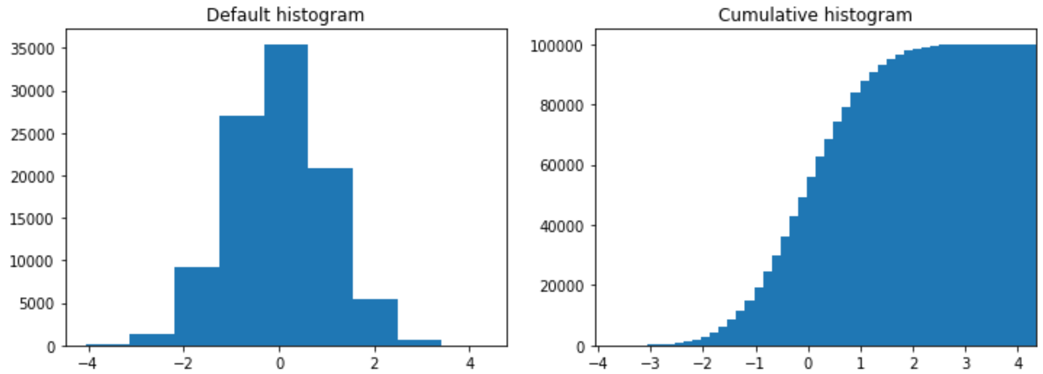
1. Use your knowledge of matplotlib to create a plot with a inset plot within it.

In [39]:
```
y = x ** 2

fig = plt.figure()

axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # inset axes


# main figure
axes1.plot(x,y,c="r")
axes1.set(title="Main_plot",ylabel='y',xlabel='x')

# inset plot
axes2.plot(y,x,c="g")
axes2.set(title="Inset_plot",ylabel='x',xlabel='y')

plt.show()
```



1. Create a cumulative histogram using the datapoints given below

In [40]:
```
# A histogram
n = np.random.randn(100000)

#Insert code for figure and subplot
fig, axes = plt.subplots(1, 2,figsize=(12,4))

#Insert code here(Default histogram)
axes[0].hist(n, bins=9)
axes[0].set(title="Default histogram")

#Insert code here(Cumulative histogram)
axes[1].hist(n, bins=50, cumulative=True)
axes[1].set(title="Cumulative histogram",xlim=(n.min(),n.max()))

plt.show()
```



Sympy

In [41]:
```
import sympy as sy
```

1. Write a few lines of code to reproduce the following number notations

In [42]:
```
x = symbols('x')

eqn = (sy.I*x+1)**2

print eqn
# Imaginary numbers

# Insert code here for imaginary numbers
```

```
(I*x + 1)**2
```

In [43]:
```
# use the given equation to replicate the answer
x = symbols('x')
eqn = ((x+11))*((x+22))*((x+33))
eqn_expand = expand(eqn)
# Insert code here
print eqn_expand
```

```
x**3 + 66*x**2 + 1331*x + 7986
```

In [44]:
```
# use the answer of the previous cell to get back the original factors
# Insert code here
print factor(eqn_expand)
```

```
(x + 11)*(x + 22)*(x + 33)
```

1. Differentiate the follwing equation: $v_0 t - \frac{1}{2} g t^2$ with respect to t and integrate the answer to get the formula back:

In [45]:
```
#Insert code here
v,g,t = symbols('v,g,t')
eqn = v*t - 0.5*g*t**2
diff_eqn = diff(eqn,t)
print diff_eqn
print integrate(diff_eqn,t)
```

```
-1.0*g*t + v
-0.5*g*t**2 + 1.0*t*v
```

In [46]:
```
#Insert code here for second derivative
print ('acceleration:')  # 2nd derivative
diff2_eqn = diff(eqn,t,2)
print diff2_eqn
#Insert code here for the integration
print integrate(integrate(diff2_eqn,t),t)
```

```
acceleration:
-1.0*g
-0.5*g*t**2
```

That's all folks!