In [ ]:
```
%pwd
```

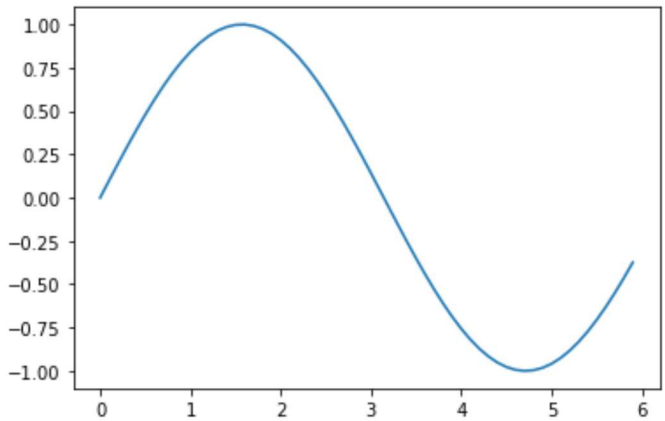
In [ ]:
```
%ls
```

In [14]:
```
# %load C:\Users\david\Documents\PYTHON_NOTEBOOKS\deep-learning-from-scratch-master\ch01\man.py
class Man:
    """サンプルクラス"""

    def __init__(self, name):
        self.name = name
        print("Initilized!")

    def hello(self):
        print("Hello " + self.name + "!")

    def goodbye(self):
        print("Good-bye " + self.name + "!")

m = Man("David")
m.hello()
m.goodbye()
```

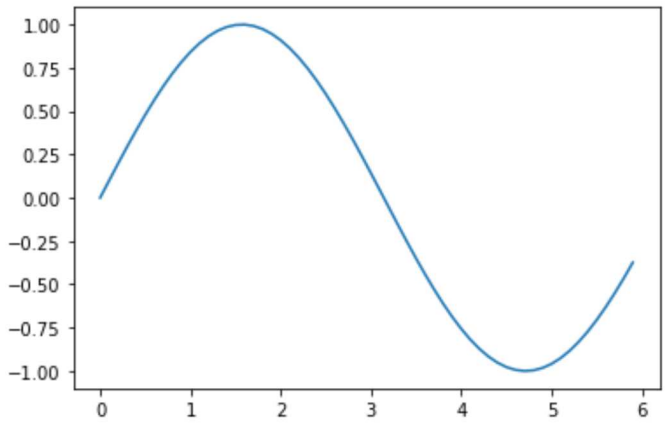
```
Initilized!
Hello David!
Good-bye David!
```

In [16]:
```
%matplotlib inline
```

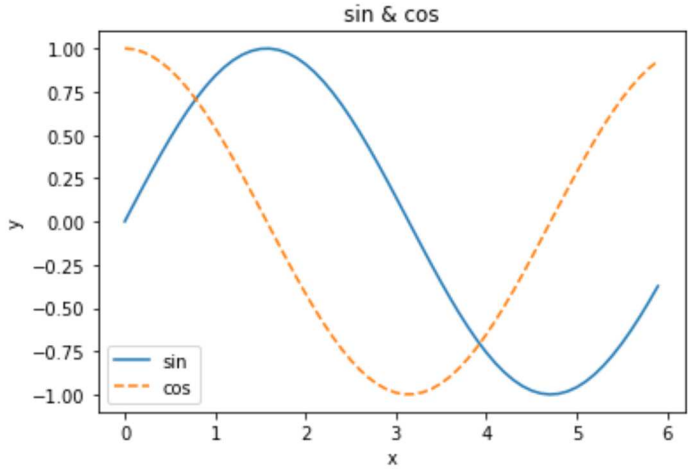
In [17]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch01/simple_graph.py
import numpy as np
import matplotlib.pyplot as plt

# データの作成
x = np.arange(0, 6, 0.1) # 0から6まで0.1刻みで生成
y = np.sin(x)

# グラフの描画
plt.plot(x, y)
plt.show()
```



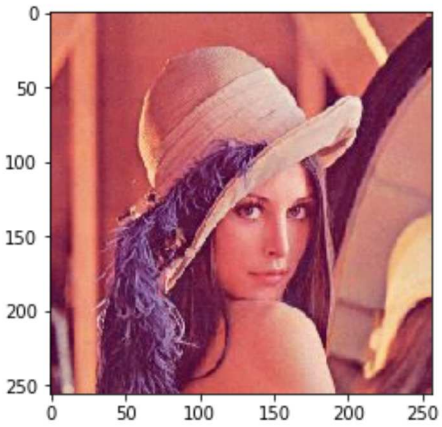
In [18]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch01/sin_graph.py
#import numpy as np
#import matplotlib.pyplot as plt

# データの作成
x = np.arange(0, 6, 0.1)
y = np.sin(x)

# グラフの描画
plt.plot(x, y)
plt.show()
```



In [19]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch01/sin_cos_graph.py
#import numpy as np
#import matplotlib.pyplot as plt

# データの作成
x = np.arange(0, 6, 0.1) # 0から6まで0.1刻みで生成
y1 = np.sin(x)
y2 = np.cos(x)

# グラフの描画
plt.plot(x, y1, label="sin")
plt.plot(x, y2, linestyle = "--", label="cos")
plt.xlabel("x") # x軸のラベル
plt.ylabel("y") # y軸のラベル
plt.title('sin & cos')
plt.legend()
plt.show()
```



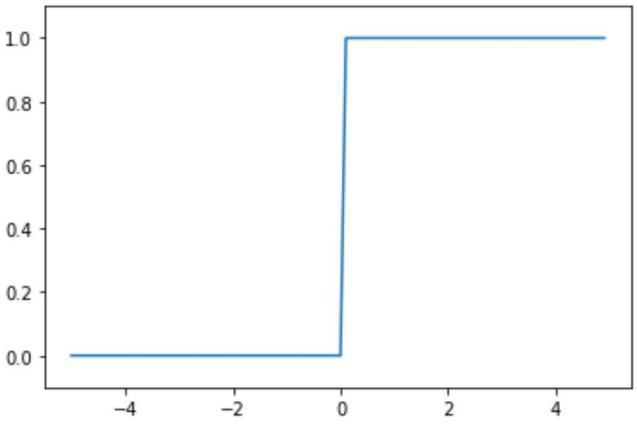
In [21]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch01/img_show.py
#import matplotlib.pyplot as plt
from matplotlib.image import imread

img = imread('dataset/lena.png') # 画像の読み込み
plt.imshow(img)

plt.show()
```



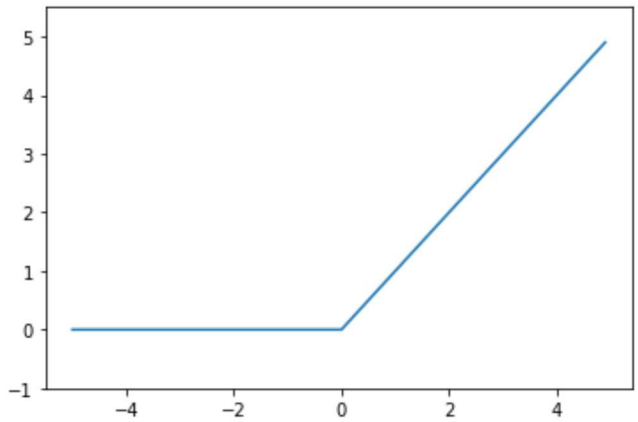
In [22]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch01/hungry.py
print("I'm hungry!")
```

```
I'm hungry!
```

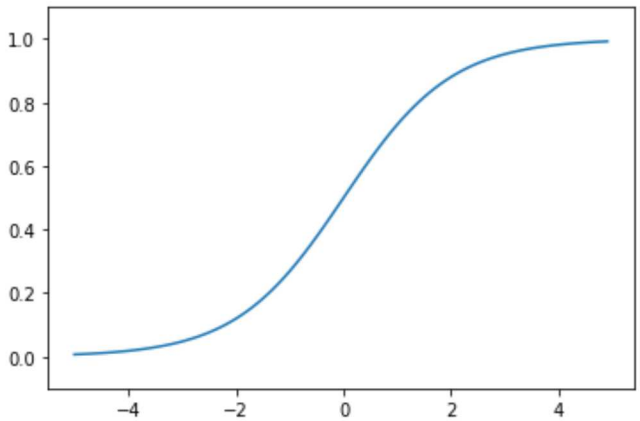
In [23]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch03/step_function.py
#import numpy as np
#import matplotlib.pylab as plt


def step_function(x):
    return np.array(x > 0, dtype=np.int)

X = np.arange(-5.0, 5.0, 0.1)
Y = step_function(X)
plt.plot(X, Y)
plt.ylim(-0.1, 1.1)   # 図で描画するy軸の範囲を指定
plt.show()
```



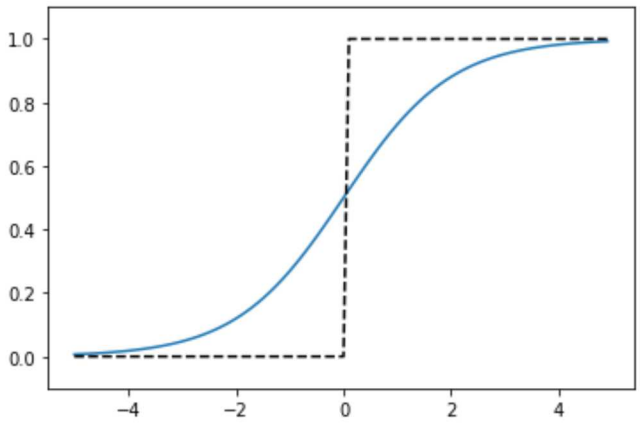
In [24]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch03/relu.py
#import numpy as np
#import matplotlib.pylab as plt


def relu(x):
    return np.maximum(0, x)

x = np.arange(-5.0, 5.0, 0.1)
y = relu(x)
plt.plot(x, y)
plt.ylim(-1.0, 5.5)
plt.show()
```



In [25]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch03/sigmoid.py
#import numpy as np
#import matplotlib.pylab as plt


def sigmoid(x):
    return 1 / (1 + np.exp(-x))

X = np.arange(-5.0, 5.0, 0.1)
Y = sigmoid(X)
plt.plot(X, Y)
plt.ylim(-0.1, 1.1)
plt.show()
```



In [26]:
```
# %load C:/Users/david/Documents/PYTHON_NOTEBOOKS/deep-learning-from-scratch-master/ch03/sig_step_compare.py
#import numpy as np
#import matplotlib.pylab as plt


def sigmoid(x):
    return 1 / (1 + np.exp(-x))


def step_function(x):
    return np.array(x > 0, dtype=np.int)

x = np.arange(-5.0, 5.0, 0.1)
y1 = sigmoid(x)
y2 = step_function(x)

plt.plot(x, y1)
plt.plot(x, y2, 'k--')
plt.ylim(-0.1, 1.1) #図で描画するy軸の範囲を指定
plt.show()
```



In [27]:
```
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else:
        return 1

if __name__ == '__main__':
    for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        y = AND(xs[0], xs[1])
        print(str(xs) + " -> " + str(y))
```

```
(0, 0) -> 0
(1, 0) -> 0
(0, 1) -> 0
(1, 1) -> 1
```

In [28]:
```
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.2
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else:
        return 1

if __name__ == '__main__':
    for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        y = OR(xs[0], xs[1])
        print(str(xs) + " -> " + str(y))
```

```
(0, 0) -> 0
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 1
```

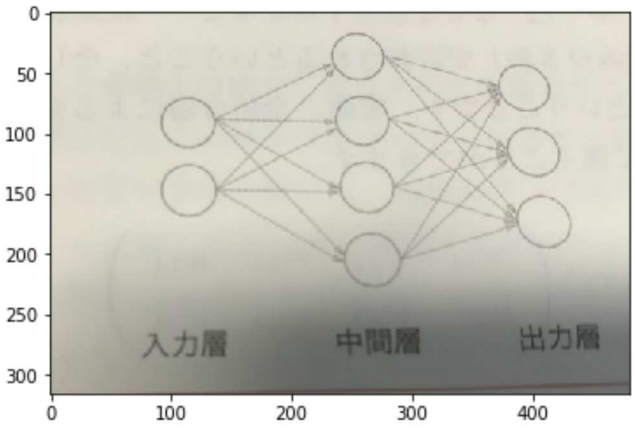
In [29]:
```
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    b = 0.7
    tmp = np.sum(w*x) + b
    if tmp <= 0:
        return 0
    else:
        return 1

if __name__ == '__main__':
    for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        y = NAND(xs[0], xs[1])
        print(str(xs) + " -> " + str(y))
```

```
(0, 0) -> 1
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 0
```

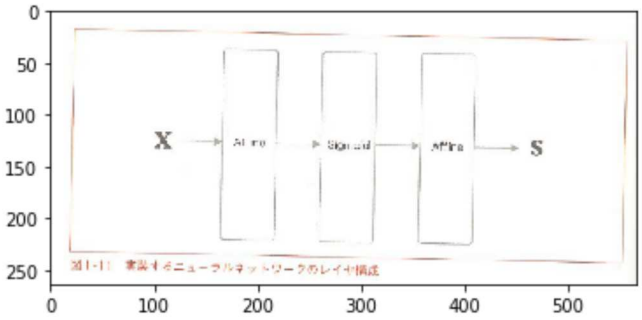
In [30]:
```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    y = AND(s1, s2)
    return y

if __name__ == '__main__':
    for xs in [(0, 0), (1, 0), (0, 1), (1, 1)]:
        y = XOR(xs[0], xs[1])
        print(str(xs) + " -> " + str(y))
```

```
(0, 0) -> 0
(1, 0) -> 1
(0, 1) -> 1
(1, 1) -> 0
```

In [35]:
```
img = imread('NN2-4-3.PNG') # 画像の読み込み
plt.imshow(img)

plt.show()
```



In [32]:
```
img = imread('F-NN.png') # 画像の読み込み
plt.imshow(img)

plt.show()
```



In [31]:
```
class Sigmoid:
    def __init__(self):
        self.params = []

    def forward(self, x):
        return 1 / (1 + np.exp(-x))


class Affine:
    def __init__(self, W, b):
        self.params = [W, b]

    def forward(self, x):
        W, b = self.params
        out = np.dot(x, W) + b
        return out


class TwoLayerNet:
    def __init__(self, input_size, hidden_size, output_size):
        I, H, O = input_size, hidden_size, output_size

        # 重みとバイアスの初期化
        W1 = np.random.randn(I, H)
        b1 = np.random.randn(H)
        W2 = np.random.randn(H, O)
        b2 = np.random.randn(O)

        # レイヤの生成
        self.layers = [
            Affine(W1, b1),
            Sigmoid(),
            Affine(W2, b2)
        ]

        # すべての重みをリストにまとめる
        self.params = []
        for layer in self.layers:
            self.params += layer.params

    def predict(self, x):
        for layer in self.layers:
            x = layer.forward(x)
        return x


x = np.random.randn(10, 2)
model = TwoLayerNet(2, 4, 3)
s = model.predict(x)
print(s)
```

```
[[ 2.05520985  0.53577627 -0.82709088]
 [ 1.83024626  0.556876   -0.73528141]
 [ 1.81327071  0.41575087 -0.6709075 ]
 [ 1.55423195  0.49925095 -0.58526596]
 [ 1.05918636  0.33765466 -0.30138932]
 [ 1.68306244  0.40609001 -0.61012937]
 [ 1.37896861  0.62572788 -0.52776512]
 [ 0.97185183  0.40805213 -0.26935227]
 [ 1.21742293  0.49720805 -0.41565873]
 [ 1.2182844   0.45872101 -0.40792756]]
```

In [ ]: