Useful Numpy Functions

In [ ]:
```
import numpy as np

#1D to 2D array
arr_1d = np.array([1, 2, 3, 4, 5, 6])
arr_2d = np.reshape(arr_1d, (2, 3))
print(arr_2d)
arr_2d = np.reshape(arr_1d, (3, 2))
print(arr_2d)
```

```
[[1 2 3]
 [4 5 6]]
[[1 2]
 [3 4]
 [5 6]]
```

In [20]:
```
arr = np.arange(12)
# Reshape to automatic rows, 4 columns
auto_rows = np.reshape(arr, (-1, 4))
print(auto_rows)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

In [21]:
```
#2D to 1D array
arr = np.array([[1, 2], [3, 4]])
arr_1d = np.reshape(arr, (1, 4))
print(arr_1d)
```

```
[[1 2 3 4]]
```

In [23]:
```
# 1. Random matrix with values between 0 and 1
random_matrix = np.random.random((3, 4))
print("Random matrix (0 to 1):")
print(random_matrix)
```

```
Random matrix (0 to 1):
[[0.72694085 0.88658461 0.3964088  0.28393973]
 [0.98743828 0.13900216 0.41515937 0.24504667]
 [0.37495154 0.48469414 0.76020205 0.62637867]]
```

In [24]:
```
# 2. Random integers matrix
int_matrix = np.random.randint(0, 100, (3, 4))
print("\nRandom integers (0-99):")
print(int_matrix)
```

```
Random integers (0-99):
[[93  1 63 57]
 [30 30 42 37]
 [61 19 32 66]]
```

In [25]:
```
# 3. Standard normal distribution (mean=0, std=1)
normal_matrix = np.random.randn(3, 4)
print("\nStandard normal distribution:")
print(normal_matrix)
```

```
Standard normal distribution:
[[ 1.81069508  0.96394854  1.29945764  0.45505495]
 [-0.80220581 -2.39139449 -0.00270885 -0.90139801]
 [ 0.85816987 -1.11497776  2.13908385 -1.00402105]]
```

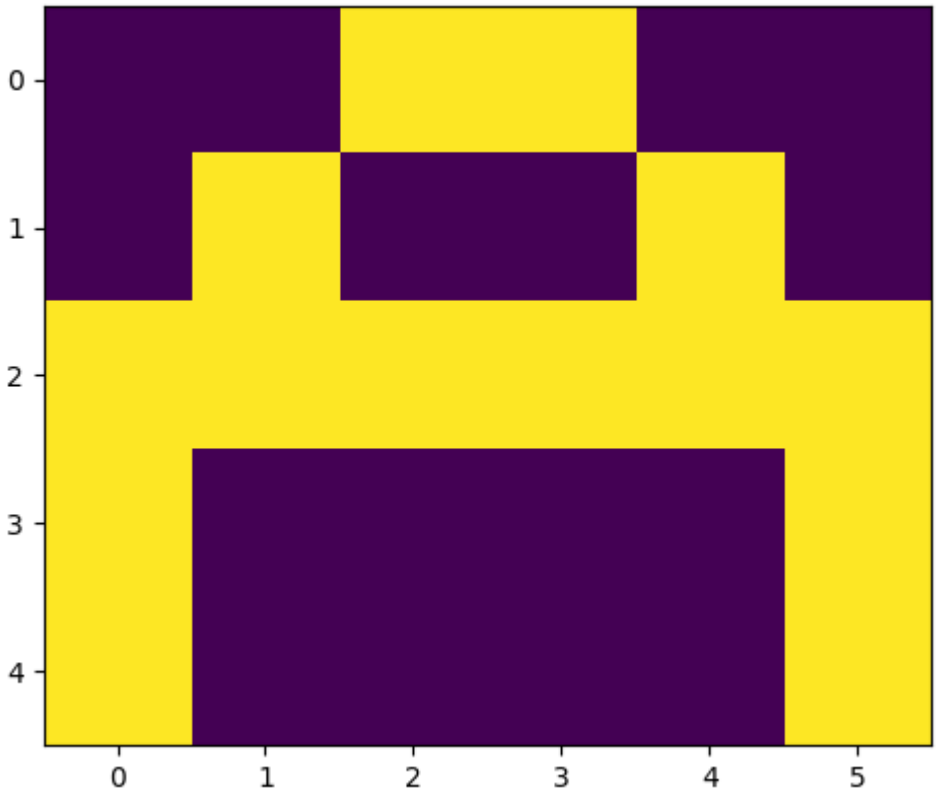
Building a Full Neural Network

In [1]:
```
# Creating data set
# A
a =[0, 0, 1, 1, 0, 0,
   0, 1, 0, 0, 1, 0,
   1, 1, 1, 1, 1, 1,
   1, 0, 0, 0, 0, 1,
   1, 0, 0, 0, 0, 1]
# B
b =[0, 1, 1, 1, 1, 0,
   0, 1, 0, 0, 1, 0,
   0, 1, 1, 1, 1, 0,
   0, 1, 0, 0, 1, 0,
   0, 1, 1, 1, 1, 0]
# C
c =[0, 1, 1, 1, 1, 0,
   0, 1, 0, 0, 0, 0,
   0, 1, 0, 0, 0, 0,
   0, 1, 0, 0, 0, 0,
   0, 1, 1, 1, 1, 0]

# Creating labels
y =[[1, 0, 0],
   [0, 1, 0],
   [0, 0, 1]]
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

# visualizing the data, plotting A.
plt.imshow(np.array(a).reshape(5, 6))
plt.show()
```

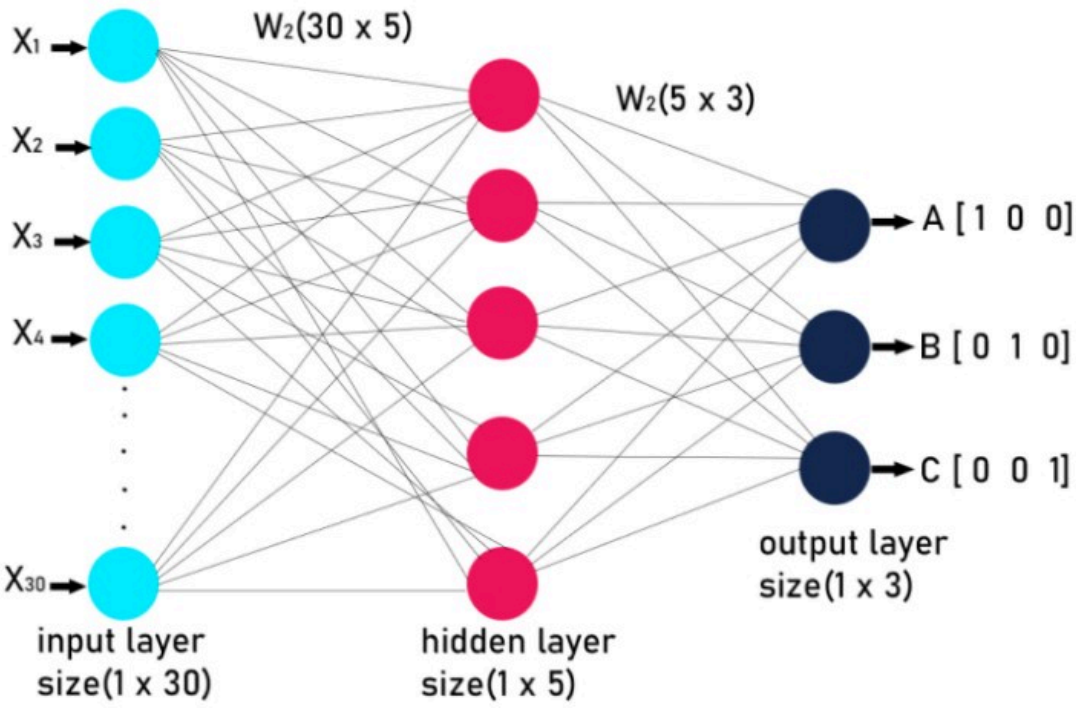


In [5]:
```
# converting data and labels into numpy array
x =[np.array(a).reshape(1, 30), np.array(b).reshape(1, 30),
                                np.array(c).reshape(1, 30)]
y = np.array(y)
```

Architecture of the Neural Network

Our neural network will have the following structure:

- Input Layer: 1 layer with 30 nodes (representing the 5x6 grid).
- Hidden Layer: 1 layer with 5 nodes.
- Output Layer: 1 layer with 3 nodes (representing the letters A, B, and C).



In [ ]:
```
# activation function
def sigmoid(x):
    #Write your code here
        return
```

In [ ]:
```
# initializing the weights randomly
w1 = #Write your code here
w2 = #Write your code here
```

In [ ]:
```
# Creating the Feed forward neural network
def f_forward(x, w1, w2):
        # hidden
        # input from layer 1
        # out put of layer 2
        # input of out layer
        # output of out layer
        return
```

In [ ]:
```
# for loss we will be using mean square error(MSE)
def loss(out, Y):
    #Write your code here
        return
```