1. How to create a class

2. how to create object of a class
3. how to define a metod

```
In [1]: # create 2 Numreical Arrays and apply dot product
        import numpy as np
```

```
In [2]: a=np.array([[2,3],[4,5]])
        b=np.array([[6,5],[4,8]])
        print(a+b)

        [[ 8  8]
         [ 8 13]]
```

```
In [3]: prod = np.dot(a,b)
```

```
In [4]: prod

Out[4]: array([[24, 34],
               [44, 60]])
```

```
In [5]: print(a*b)

        [[12 15]
         [16 40]]
```

```
In [8]: def hello():
            print("welcome to MCA")
```

```
In [9]: hello()

        welcome to MCA
```

```
In [10]: def funct(x):
             if(x%2 == 0):
                 print("EVEN")
             else:
                 print("ODD")
```

```
In [11]: funct(12)

         EVEN
```

```
In [13]: sub=["java","python","c++","c"]
         for x in sub:
             print (x)

         java
         python
         c++
         c
```

In [14]:

```
for x in "HIRAY":
    print(x)
```

```
H
I
R
A
Y
```

In [15]:

```
for x in range(10):
    print(x)
```

```
0
1
2
3
4
5
6
7
8
9
```

class

In [16]:

```
class Student:
    name="XYZ"
    marks=20
```

In [17]:

```
stud=Student()
```

In [18]:

```
print(stud.name)
```

```
XYZ
```

In [19]:

```
print(stud.marks)
```

```
20
```

In [20]:

```
class Subjects:
    def __init__(self,name,marks):
        self.name=name
        self.marks=marks
```

In [21]:

```
s1 = Subjects("Java",35)
```

In [22]:

```
print(s1.name)
```

```
Java
```

In [23]:

```
print(s1.marks)
```

```
35
```

In [25]:
```
# enumerate in python
x=["java","python","c++"]
y=enumerate(x)
print(x)
```

```
['java', 'python', 'c++']
```

In [27]:
```
x=["java","python","c++"]
for i, n in enumerate(x):
    print(n)
```

```
java
python
c++
```

Adaline

In [28]:
```
import numpy as np
```

In [29]:
```
class Adaline:
    def __init__(self,input_size,learning_rate=0.1,epochs=100):
        self.weights=np.zeros(input_size)
        self.bias=0;
        self.learning_rate=learning_rate
        self.epochs=epochs
    def activation(self,X):# X={x1,x2...xn}
        return X
    def predict(self,X):# net input
        return self.activation(np.dot(X,self.weights)+self.bias)
    def train(self,X,y):
        for epoch in range(self.epochs):
            for i in range(len(X)):
                prediction=self.predict(X[i])
                error=y[i]-prediction
                # update Weights and bias
                self.weights+=self.learning_rate*error*X[i]
                self.bias+=self.learning_rate*error
    def evaluate(self,X):
        return np.where(self.predict(X)>=0.5,1,0)
```

In [30]:
```
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0,0,0,1])
```

In [31]:
```
adaline=Adaline(input_size=2,learning_rate=0.1,epochs=100)
adaline.train(X,y)
predictions=adaline.evaluate(X)
```

In [32]:
```
print(predictions)
```

```
[0 0 0 1]
```

In [35]:

```
for i, p in enumerate(predictions):
    print(f"Input:{X[i]} => Predicted:{p}=>Actual:{y[i]}")
```

```
Input:[0 0] => Predicted:0=>Actual:0
Input:[0 1] => Predicted:0=>Actual:0
Input:[1 0] => Predicted:0=>Actual:0
Input:[1 1] => Predicted:1=>Actual:1
```

In [ ]: