Implementation of a two-layer neural network

1. import

2. load dataset, set X and Y, shape of X and Y

3. def layer\_sizes(X, Y) => n\_x, n\_y

def initialize\_parameters(n\_x, n\_hidden, n\_y) => parameters{w1, b1, w2, b2}

def forward\_propagation(X, parameters) => cache{Z1, A1, Z2, A2}

def compute\_cost(A2, Y, parameters) => cost

def backward\_propagation(parameters, cache, X, Y) => grads{dw1, db1, dw2, db2}

def update\_parameters(parameters, grads, learning\_rate) => parameters{w1, b1, w2, b2}

def nn\_model(X, Y, n\_h, num\_iterations, print\_cost) => parameters{w1, b1, w2, b2}

n\_x, n\_y = layer\_sizes

parameters = initialize\_parameters

W1, b1, W2, b2

for i in range(0, num\_iterations):

cache = forward\_propagation

cost = compute\_cost

grads = backward\_propagation

parameters = update\_parameters

return parameters

def predict(parameters, X) => predictions