

RED NEURONAL DE 3 CAPAS

INTEGRANTES:

Morelos Lira Jose Luis

Olivares Cordero Víctor Hugo



HERRAMIENTAS

Python 3.6

IDE Pycharm

Numpy python

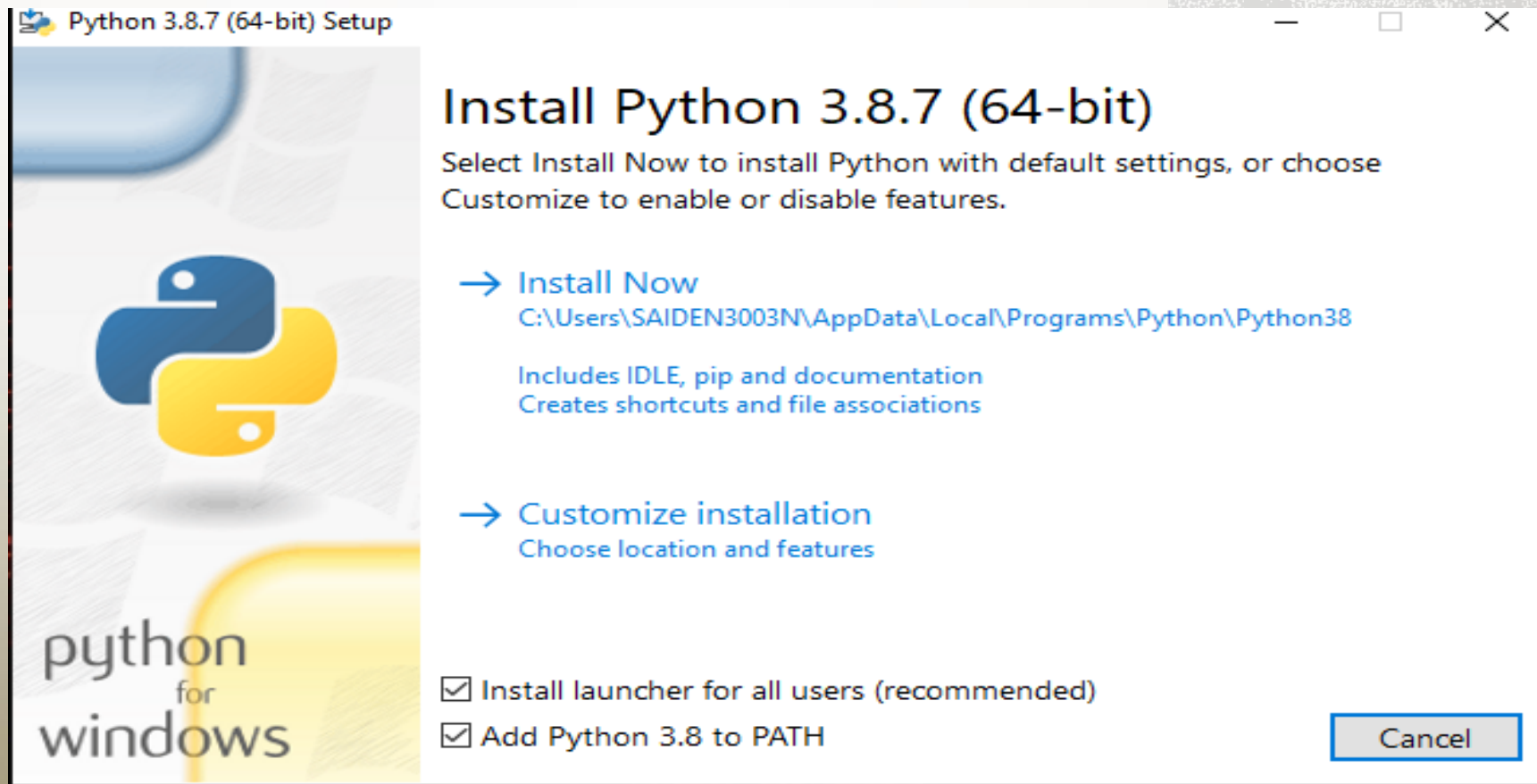


INSTALACIÓN DE LAS HERRAMIENTAS



PYTHON

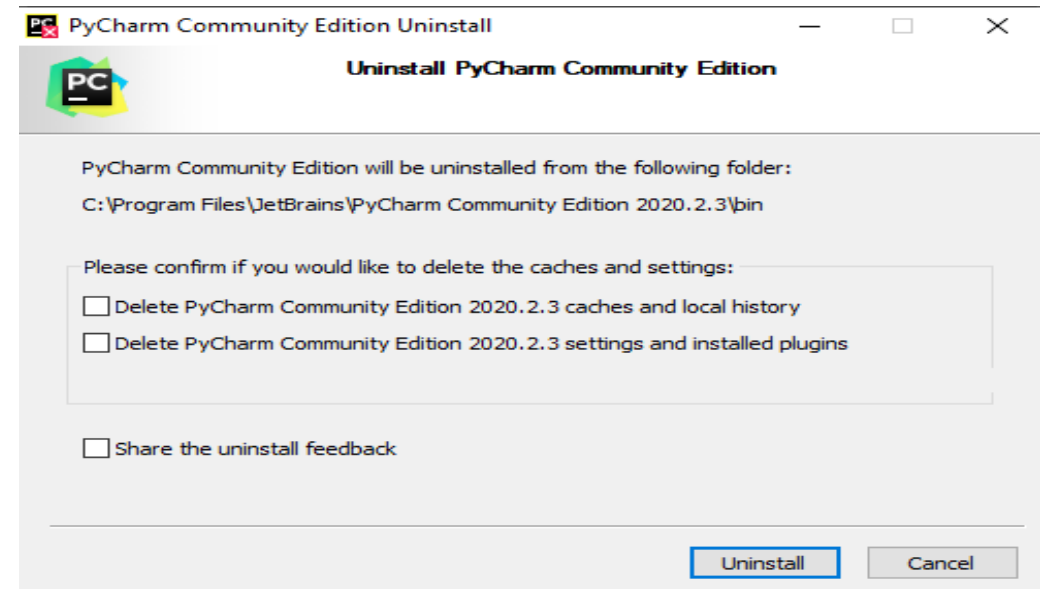
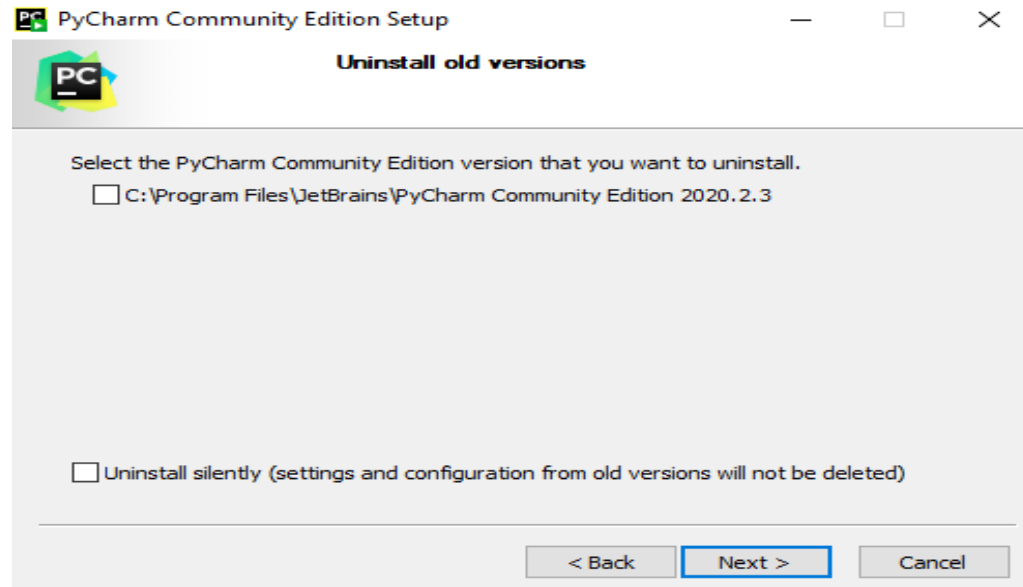
- Para instalar Python por el archivo ejecutable que se descarga de su pagina oficial.
- se seleccionaría las dos opciones agregando al path



PYCHARM

**SE LE OTORGARA PERMISOS DE
USUARIOS**

**AL IGUAL QUE PYTHON SE
AGREGARA AL PATH**



CREACIÓN DEL ENTORNO Y DESCARGA DE LIBRERÍAS



Se instalaran la Librería numpy para el funcionamiento del programa

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19042.746]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.

C:\Users\josep>pip numpy
ERROR: unknown command "numpy"

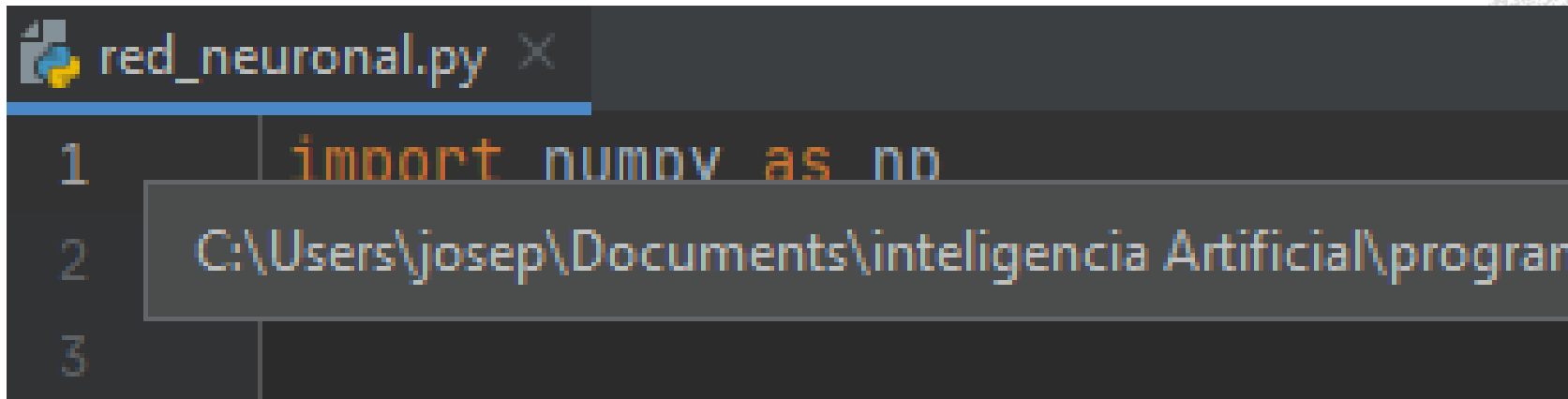
C:\Users\josep>pip install numpy
Requirement already satisfied: numpy in c:\users\josep\appdata\local\programs\python\python37-32\lib\site-packages (1.19.5)

C:\Users\josep>_
```



IMPORTAR LAS LIBRERIAS

- se Importa la Libreria Numpy para el funcioamiento correcto del programa , esta libreria sera necesaria y la mayor parte del funcionamiento es gracias a esta



```
red_neuronal.py X
1 import numpy as np
2 C:\Users\josep\Documents\inteligencia Artificial\programa
3
```



Las funciones que contiene el programa

```
def sigmoid(x):  
    return 1.0 / (1.0 + np.exp(-x))  
  
def sigmoid_derivada(x):  
    return sigmoid(x) * (1.0 - sigmoid(x))  
  
def tanh(x):  
    return np.tanh(x)  
  
def tanh_derivada(x):  
    return 1.0 - x ** 2
```

Que son instanciadas a la clase neurolink

```
20 class NeuralNetwork:  
21  
22     def __init__(self, layers, activation='tanh'):  
23         if activation == 'sigmoid':  
24             self.activation = sigmoid  
25             self.activation_prime = sigmoid_derivada  
26         elif activation == 'tanh':  
27             self.activation = tanh  
28             self.activation_prime = tanh_derivada  
29  
30         # inicializo los pesos  
31         self.weights = []  
32         self.deltas = []  
33         # capas = [2,3,2]  
34         # rando de pesos varia entre (-1,1)  
35         # asigno valores aleatorios a capa de entrada y capa oculta  
36         for i in range(1, len(layers) - 1):  
37             r = 2 * np.random.random((layers[i - 1] + 1, layers[i] + 1)) - 1  
38             self.weights.append(r)  
39         # asigno aleatorios a capa de salida  
40         r = 2 * np.random.random((layers[i] + 1, layers[i + 1])) - 1  
41         self.weights.append(r)  
42
```



se creara un for para entrar al contenido en los tags para los patrones y acceder a ellos, el contenido de ello se almacenara en auxpalabra, junto con nltk.word_tokenize para reconocer los signos especiales.

En auxY se guardaran los tags repetido para pasarlos a individual se usa un if.

```
palabras=[]
tags=[]
auxX=[]
auxY=[]

for contenido in datos["contenido"]:
    for patrones in contenido["patrones"]:
        auxPalabra = nltk.word_tokenize(patrones)
        palabras.extend(auxPalabra)
        auxX.append(auxPalabra)
        auxY.append(contenido["tag"])

        if contenido["tag"] not in tags:
            tags.append(contenido["tag"])

print(palabras)
print(auxX)
print(auxY)
print(tags)
```

```
['hola'], ['un', 'saludo'], ['hello'], ['buenos', 'dias'], ['postre'], ['que', 'postre', 'tienes'], ['que', 'hay', 'de', 'postre'], ['que', 'pontres', 'hay'], ['adios'], ['hasta', 'luego'], ['nos', 'vemos'], ['hasta', 'la', 'proxima'], ['abierto'], ['abren'], ['a', 'que', 'hora', 'abren'], ['desde', 'a', 'que', 'hora', 'estan', 'abiertos'], ['a', 'que', 'hora', 'aceptan', 'clientes'], ['sin', 'reservacion'], ['aceptan', 'clientes', 'sin', 'reservacion'], ['cerrado'], ['cierran'], ['a', 'que', 'hora', 'cierran'], ['cuantas', 'personas', 'son', 'en', 'la', 'reservacion'], ['cuantas', 'personas', 'por', 'reservacion'], ['cuantos', 'comensales', 'pueden', 'ir', 'en', 'una', 'reservacion'], ['cuantas', 'veces', 'puedo', 'reservar'], ['puedo', 'hacer', 'mas', 'de', 'una', 'reservacion'], ['puedo', 'hacer', 'mas', 'reservaciones'], ['pueden', 'entrar', 'niños'], ['pueden', 'entrar', 'bebes'], ['cuantos', 'niños', 'pueden', 'ir', 'por', 'reservacion'], ['como', 'estas'], ['hola', 'como', 'estas'], ['que', 'tal', 'estas']]
['saludo', 'saludo', 'saludo', 'saludo', 'postre', 'postre', 'postre', 'postre', 'despedida', 'despedida', 'despedida', 'despedida', 'abierto', 'abierto', 'abierto', 'abierto', 'abierto', 'abierto', 'sin', 'sin', 'cerrado', 'cerrado', 'cerrado', 'personas', 'personas', 'personas', 'reservacion', 'reservacion', 'reservacion', 'niños', 'niños', 'niños', 'estas', 'estas', 'estas']
['saludo', 'postre', 'despedida', 'abierto', 'sin', 'cerrado', 'personas', 'reservacion', 'niños', 'estas']

(chat) C:\Users\SAIDEN3003N\chat>
```



Se usara la línea:

Una función para predecir

Junto a otras dos que serán las que nos impriman por último de manda a llamar una variable donde mandamos a instanciar la clase que nos devolverá el resultado dándole los parámetros a nuerolink

```
def predict(self, x):
    ones = np.atleast_2d(np.ones(x.shape[0]))
    a = np.concatenate((np.ones(1).T, np.array(x)), axis=0)
    for l in range(0, len(self.weights)):
        a = self.activation(np.dot(a, self.weights[l]))
    return a

def print_weights(self):
    print("LISTADO PESOS DE CONEXIONES")
    for i in range(len(self.weights)):
        print(self.weights[i])

def get_deltas(self):
    return self.deltas

# funcion XOR
nn = NeuralNetwork([2,2,1])
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])
y = np.array([0, 1, 1, 0])
nn.fit(X, y, epochs=2000)
for e in X:
    print("Entradas:", e, "Salidas:", nn.predict(e))
```



Para poder hacer funcionar la red neuronal lo que se hace es mandar a llamar la clase que tiene las instanciación de las clase y la funciones esto lo hacemos mediante una variable para poder manipular los datos , al igual que le damos unos valores , rango con lo que trabajara la función de salida es una función booleana una función xor

```
# funcion XOR
nn = NeuralNetwork([2,2,1])
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])
y = np.array([0, 1, 1, 0])
nn.fit(X, y, epochs=2000)
for e in X:
    print("Entradas:" , e , "Salidas:" , nn.predict(e))
```



```
red_neuronal x
C:\Users\josep\AppData\Local\Programs\Python\Python37-32\python.exe "C:/Users/josep/Documents/intelig
epochs: 0
Entrdas: [0 0] Salidas: [0.00218331]
Entrdas: [0 1] Salidas: [0.96934895]
Entrdas: [1 0] Salidas: [0.96153521]
Entrdas: [1 1] Salidas: [0.00767215]

Process finished with exit code 0
```

4: Run TODO 6: Problems Terminal Python Console R Console

7 has been configured as the project interpreter // Configure a Python Interpreter... (7 minutes ago) 85:46

Son los resultados del programa como se puede visualizar lo manejamos como entradas y salida haciendo el uso de un arreglo unidimensional para mostrar los resultados

