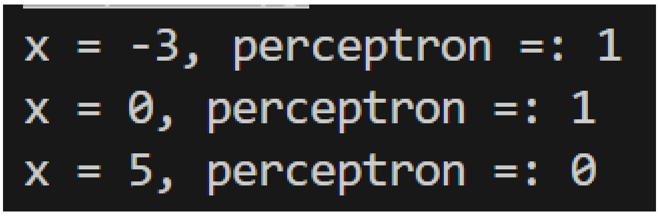
```
import numpy as np

def affine(b, w, x):
    return np.dot(w, x) + b

def activation(s):

   return 1 if s >= 0 else 0

def relu(s):
    return max(0, s)

def heaviside(s):

return 1 if s >= 0 else 0

def perceptron(b, w, x):
    s = affine(b, w, x)
    return
heaviside(s)

def perceptron_ou(x1, x2):
    b = -0.5
    w = np.array([1, 1])
    x =
np.array([x1, x2])
    return perceptron(b, w, x)

def perceptron_et(x1, x2):
    b =
-1.5
    w = np.array([1, 1])
    x = np.array([x1, x2])
    return perceptron(b, w,
x)

def perceptrons(b, w, x, activation_function):
    s = affine(b, w, x)
    return
activation_function(s)

def neural_network(x, activation_function=relu):
    w1 =
np.array([[1, -1], [-1, 2]])
    b1 = np.array([1, -1])
    layer1_output =
[perceptrons(b, w, x, activation_function) for b, w in zip(b1, w1)]

    w2 =
np.array([1, 1])
    b2 = -1
    layer2_output = perceptrons(b2, w2,
layer1_output, activation_function)

    return layer2_output

def network_and(x1,
x2):
    w1 = np.array([1, 1])
    b1 = -1.5
    x = np.array([x1, x2])

return perceptron(b1, w1, x)

def network_other(x1, x2):
    w1 = np.array([1, 1])
    b1
= -0.5
    x = np.array([x1, x2])
    return perceptron(b1, w1, x)

def
neural_network_xor(inputs):
    w1 = np.array([1, -1])
    b1 = -0.5
    output1 =
perceptron(b1, w1, inputs)

    w2 = np.array([-1, 1])
    b2 = -0.5
    output2 =
perceptron(b2, w2, inputs)

    w_out = np.array([1, 1])
    b_out = -1
    final_output =
perceptron(b_out, w_out, np.array([output1, output2]))

    return final_output

#Exo 1
b
= 0.5
w1 = -2/3
x_values = [-3, 0, 5]

for x in x_values:
    result = perceptron(b, w1,
x)
    print(f"x = {x}, perceptron =: {result}")

#Exo 2

b = 0
w2 = -1
x
= -0.3
result = perceptron(b, w2, x)
if(result == 1):
    print("carre")
else :

   print("rond")

#Exo 3

inputs = [(0, 0), (0, 1), (1, 0), (1,
1)]

print("Perceptron OU:")
for x1, x2 in inputs:
    print(f"OU({x1},
{x2}) = {perceptron_ou(x1, x2)}")

print("\nPerceptron ET:")
for x1, x2 in
inputs:
    print(f"ET({x1}, {x2}) = {perceptron_et(x1, x2)}")

#Exo
4

test_inputs = [(4, 7), (3, 2), (0.1, 0.1)]

for x, y in test_inputs:
    x_input =
np.array([x, y])
    result = neural_network(x_input)
    print(f"Entrée: ({x}, {y})
-> Sortie: {result}")

#Exo 5

inputs = [(0, 0), (0, 1), (1, 0), (1,
1)]

print("Réseau ET logique:")
for x1, x2 in inputs:
    print(f"ET({x1},
{x2}) = {network_and(x1, x2)}")

print("\nRéseau autre opérateur:")
for x1,
x2 in inputs:
    print(f"Opérateur({x1}, {x2}) = {network_other(x1, x2)}")


#Exo 6

inputs = [(0, 0), (0, 1), (1, 0), (1, 1)]

for inp in inputs:
    result =
neural_network_xor(inp)
    print(f"Entrées {inp}, Sortie {result}")
```

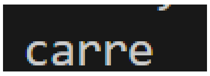
résultat

exo 1



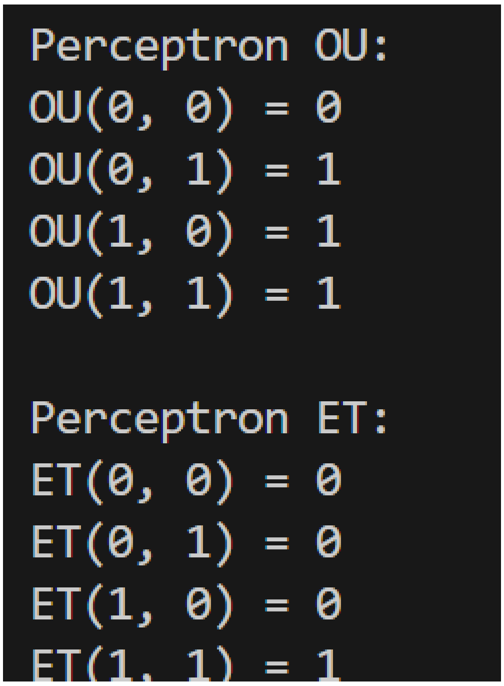
```
x = -3, perceptron =: 1
x = 0, perceptron =: 1
x = 5, perceptron =: 0
```

Exo 2



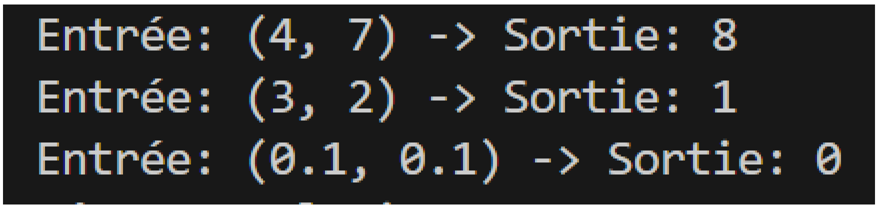
```
carre
```

Exo 3



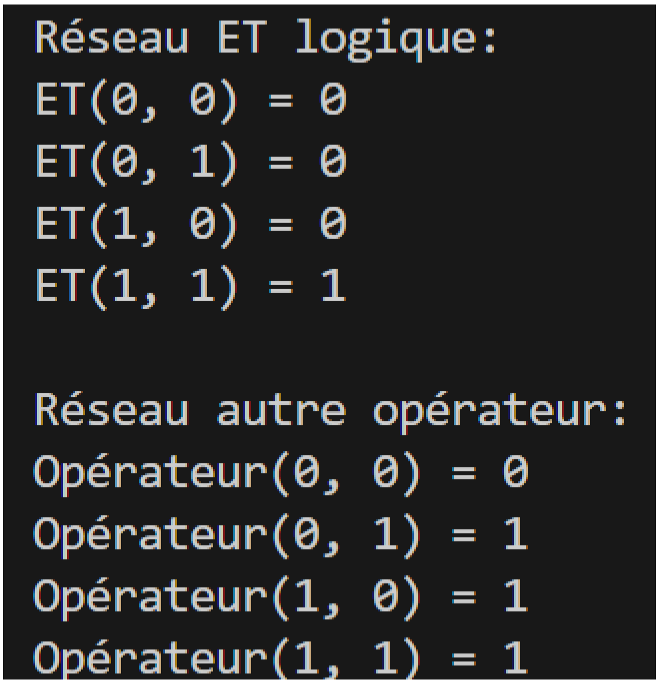
```
Perceptron OU:
OU(0, 0) = 0
OU(0, 1) = 1
OU(1, 0) = 1
OU(1, 1) = 1

Perceptron ET:
ET(0, 0) = 0
ET(0, 1) = 0
ET(1, 0) = 0
ET(1, 1) = 1
```

Exo 4



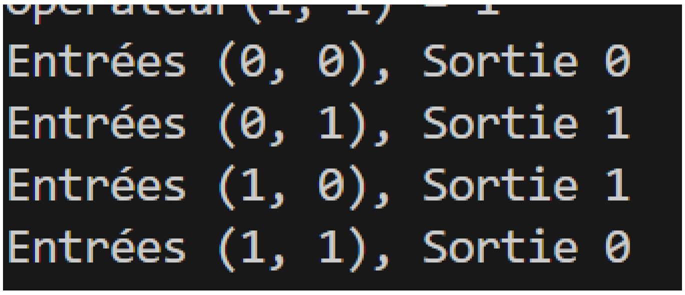
```
Entrée: (4, 7) -> Sortie: 8
Entrée: (3, 2) -> Sortie: 1
Entrée: (0.1, 0.1) -> Sortie: 0
```

Exo 5



```
Réseau ET logique:
ET(0, 0) = 0
ET(0, 1) = 0
ET(1, 0) = 0
ET(1, 1) = 1

Réseau autre opérateur:
Opérateur(0, 0) = 0
Opérateur(0, 1) = 1
Opérateur(1, 0) = 1
Opérateur(1, 1) = 1
```

Exo 6



```
Opérateur(1, 1) = 1
Entrées (0, 0), Sortie 0
Entrées (0, 1), Sortie 1
Entrées (1, 0), Sortie 1
Entrées (1, 1), Sortie 0
```