In dit document los ik deze vraag op:

Initialiseer een `Neuron` voor elk van de INVERT-, AND- en OR-poorten **met dezelfde parameters als bij de** `Percepton` . Verklaar waarom dit (niet) werkt? Als het niet goed werkt, initialiseer de `Neuron` dan met andere parameters, zodat de poorten wel correct functioneren.

De tuples/lists zijn inputs en de losse getallen rechts daarvan zijn de outputs, we gebruiken hier de sigmoid functie afgerond met de standaard python round functie:

```
AND(p1 parameters):
(0, 0) 0
(0, 1) 0
(1, 0) 0
(1, 1) 0
OR(p1 parameters):
(0, 0) 0
(0, 1) 0
(1, 0) 0
(1, 1) 1
INVERT(p1 parameters):
[1] 0
[0] 1
```

Zoals je kan zien werken de OR en AND gate niet correct, de INVERT gate blijft echter wel werken. Waarom? Dat kunnen uitzoeken we door het te berekenen. Laten we beginnen met wat er in de AND neuron niet kopt, [1,1]:

Input = [1,1]

Weights = [1,1]

Bias = -2

1*1 + 1*1 -2 = 0

Sigmoid(0) = 0.5

Rounded_sigmoid = 0

.....

Waarom krijgen we niet 1 als afgerond getal terug? Kennelijk rond de python round functie net op een wat andere manier af, op .5 naar het dichtsbezijnde EVEN getal.

Om dit op te lossen importen we decimal:

```
def rounded_sigmoid(x):

    # the standard python round doesn't always round op x.5
    with localcontext() as ctx:
        ctx.rounding = ROUND_HALF_UP
        return int(Decimal(sigmoid(x)).to_integral_value())
```

Nu wordt .5 wel altijd naar boven afgerond. Dit is ook terug te zien in onze tests:

```
AND(p1 parameters):
(0, 0) 0
(0, 1) 0
(1, 0) 0
(1, 1) 1
OR(p1 parameters):
(0, 0) 0
(0, 1) 1
(1, 0) 1
(1, 1) 1
INVERT(p1 parameters):
[1] 0
[0] 1
```

Dus ja, het gebruiken van de perceptron parameters voor sigmoid neurons werkt! (als je maar goed afrond)