```
import numpy as np
np.random.seed(seed=0)
I = np.random.choice([0,1,0], 3)
W = np.random.choice([-1,1,1], 3)
print(f'Input vector:{I}, Weight vector:{W}')
```

```
Input vector:[0 1 0], Weight vector:[1 1 1]
```

```
dot = I @ W
print(f'Dot product: {dot}')
```

```
Dot product: 1
```

```
def linear_threshold_gate(dot: int, T: float) -> int:
    '''Returns the binary threshold output'''
    if dot >= T:
        return 1
    else:
        return 0
```

```
T = 1
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
```

```
Activation: 1
```

```
T = 3
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
```

```
Activation: 0
```

```
input_table = np.array([
    [0,0,0],
    [0,0,1],
    [0,1,0],
    [0,1,1],
    [1,0,0],
    [1,0,1],
    [1,1,0],
    [1,1,1]
])

print(f'input table:\n{input_table}')
```

```
input table:
[[0 0 0]
 [0 0 1]
 [0 1 0]
 [0 1 1]
 [1 0 0]
 [1 0 1]
 [1 1 0]
 [1 1 1]]
```

OR

```
weights = np.array([1,1,1])
print(f'weights: {weights}')
```

```
weights: [1 1 1]
```

```
dot_products = input_table @ weights
print(f'Dot products: {dot_products}')
```

```
Dot products: [0 1 1 2 1 2 2 3]
```

```
T = 1
for i in range(0,8):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

```
Activation: 0
Activation: 1
Activation: 1
```

```
Activation: 1
Activation: 1
Activation: 1
Activation: 1
Activation: 1
```

AND

```
weights = np.array([1,1,1])
print(f'weights: {weights}')
```

```
weights: [1 1 1]
```

```
dot_products = input_table @ weights
print(f'Dot products: {dot_products}')
```

```
Dot products: [0 1 1 2 1 2 2 3]
```

```
T = 3
for i in range(0,8):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

```
Activation: 0
Activation: 0
Activation: 0
Activation: 0
Activation: 0
Activation: 0
Activation: 0
Activation: 1
```

Tautology

```
T = 0
for i in range(0,8):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

```
Activation: 1
Activation: 1
Activation: 1
Activation: 1
Activation: 1
Activation: 1
Activation: 1
Activation: 1
```

NOT

```
input_table1 = np.array([
    [0], # both no
    [0], # one no, one yes
    [1], # one yes, one no
    [1]  # bot yes
])

print(f'input table:\n{input_table}')
```

```
input table:
[[0]
 [0]
 [1]
 [1]]
```

```
weights = np.array([-1])
print(f'weights: {weights}')
```

```
weights: [-1]
```

```
dot_products = input_table1 @ weights
print(f'Dot products: {dot_products}')
```

```
Dot products: [ 0  0 -1 -1]
```

```
T = 0
for i in range(0,4):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

```
Activation: 1
Activation: 1
Activation: 0
Activation: 0
```

NOTAND

```
weights = np.array([1,1,1])
print(f'weights: {weights}')
```

```
weights: [1 1 1]
```

```
dot_products = input_table @ weights
print(f'Dot products: {dot_products}')
```

```
Dot products: [0 0 0 0 1 1 1 1]
```

```
T = 1
for i in range(0,8):
    activation = linear_threshold_gate(dot_products[i], T)
    print(f'Activation: {activation}')
```

```
Activation: 0
Activation: 0
Activation: 0
Activation: 0
Activation: 1
Activation: 1
Activation: 1
Activation: 1
```