【要点まとめ】 入力された各ノードに対して、重要度をもとに重み(w)を決定する。

たとえば一次関数であれば、wは傾き、bは切片である。 傾きと切片を上手く求めるのがNNの重要な目的の一つとなる。

入力(x) * 重み(w) + バイアス(b) は python式で np.dot(x,w)+b で表される。（dotは行列演算）

【実装演習】 ※ print_vecは出力用の独自関数

```
# 重み
W = np.array([[1,3,5],[2,4,6]])
print_vec("重み", W)

# バイアス
b = np.array([0.5,-0.6, 0.7])
print_vec("バイアス", b)

# 入力値
x = np.array([3, 4])
print_vec("入力", x)

# 総入力
u = np.dot(x, W) + b
print_vec("総入力", u)
```

```
*** 重み ***
[[1 3 5]
 [2 4 6]]
shape: (2, 3)

*** バイアス ***
[ 0.5 -0.6  0.7]
shape: (3,)

*** 入力 ***
[3 4]
shape: (2,)

*** 総入力 ***
[11.5 24.4 39.7]
shape: (3,)
```

※ 参考資料：ゼロから作るDeepLearning

【実装演習考察】 単純なNNの仕組みを実装するのは簡単だと思いましたが、計算式の次元数を一致させるなど 入力・重み・バイアスがどこに当たるかをキチンと理解しておく必要がありました。