In [1]:

```
import numpy as np
from numpy.typing import ArrayLike
```

In [2]:

```
def dot(A:ArrayLike, B:ArrayLike)->ArrayLike:
    '''
    Parameters
    ----------
    A: numpy.typing.ArrayLike
        Numpy Array for multiplication

    B: numpy.typing.ArrayLike
        Numpy Array for multiplication

    Returns
    ----------
    res: numpy.typing.ArrayLike
        Numpy Array representing the dot product of A and B
    '''

    # extracting the rows and columns from the matrices
    r1, c1 = A.shape
    r2, c2 = B.shape

    # validating the operation
    assert c1 == r2, \
    f"Invalid operation: Num cols of A and Num rows of B are not equal for {A.shape}, {B.s

    # creating an array of zeros
    res = np.zeros((r1,c2), dtype=A.dtype)

    # populating res
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                res[i][j] += int(A[i][k]) * int(B[k][j])
    return res
```

In [3]:

```
# a fun little list comprehension solution

def dot_comprehension(A:ArrayLike, B:ArrayLike)->ArrayLike:
    '''
    Parameters
    ----------
    A: numpy.typing.ArrayLike
        Numpy Array for multiplication

    B: numpy.typing.ArrayLike
        Numpy Array for multiplication

    Returns
    ----------
    res: numpy.typing.ArrayLike
        Numpy Array representing the dot product of A and B
    '''

    # extracting the rows and columns from the matrices
    r1, c1 = A.shape
    r2, c2 = B.shape

    assert c1 == r2, \
    f"Invalid operation: Num cols of A and Num rows of B are not equal for {A.shape}, {B.s
```

```
    res = np.array([[sum(a*b for a,b in zip(A_row,B_col)) for B_col in zip(*B)] for A_row
    return res
```

In [4]:
```
A = np.array([[-4,-3,-2], [6,0,-1], [2,1,3]])
B = np.array([[5,4], [6,7],[-4,-3]])

X = np.array([[1,0],[0,1]])
Y = np.array([[1,2,3,4],[5,6,7,8]])
```

In [5]:
```
print(f"Dot:\n{dot(A,B)}\n\nNumpy Dot:\n{np.dot(A,B)}")
```

```
Dot:
[[-30 -31]
 [ 34  27]
 [  4   6]]

Numpy Dot:
[[-30 -31]
 [ 34  27]
 [  4   6]]
```

In [6]:
```
print(f"Dot:\n{dot(X,Y)}\n\nNumpy Dot:\n{np.dot(X,Y)}")
```

```
Dot:
[[1 2 3 4]
 [5 6 7 8]]

Numpy Dot:
[[1 2 3 4]
 [5 6 7 8]]
```

In [ ]:
```

```