Matrix multiplication in Python vs. NumPy

```
# In Python, matrices can be implementd as lists of lists.
A = [[1, -1, 2],
     [0, 3, 1]]
B = [[1, 2],
     [3, 0],
     [2, 1]]

# Python funciton for multiplying two matrices.
def mat_mul(A, B):
  rowsA = len(A)
  colsA = len(A[0])
  rowsB = len(B)
  colsB = len(B[0])

  if colsA != rowsB:
    print("Cannot multiply these matrices, go learn some linear algebra!")
    return None

  C = []
  for i in range(rowsA):
    C.append([]) # create the next (empty as of now) row i
    for j in range(colsB):
      # Compute C[i][j]
      cij = 0
      for k in range(colsA):
        cij = cij + A[i][k] * B[k][j]
      C[i].append(cij) # this sets C[i][j] = cij

  return C
```

```
mat_mul(A, B)
```

```
[[2, 4], [11, 1]]
```

NumPy version

```
import numpy as np

# NumPy can multiply Python matrices represented as lists of lists.
np.dot(A, B)
```

```
array([[ 2,  4],
       [11,  1]])
```

```
np.matmul(A, B)
```

```
array([[ 2,  4],
       [11,  1]])
```

```
# Let's create matrices as NumPy arrays.
npA = np.array(A)
npB = np.array(B)
print(npA)
print(npB)
print(type(npA))
```

```
[[ 1 -1  2]
 [ 0  3  1]]
[[1 2]
 [3 0]
 [2 1]]
<class 'numpy.ndarray'>
```

```
# Use the @ operator to multiply two matrices (Arrays) in NumPy.
npA @ npB
```

```
array([[ 2,  4],
       [11,  1]])
```