Lab Analisis Data 1 - Modul 2

MATH2031 Aljabar Linear

Calvin Institute of Technology

Capaian Pembelajaran:

Setelah mengikuti modul ini, mahasiswa akan dapat melakukan proses operasi matriks dasar dengan Python dan NumPy

1 Variabel Matriks dengan Library numpy

Numpy adalah suatu modul ekstensi Python yang sering dipakai oleh para ilmuwan data *(data scientist)*. Numpy menyediakan banyak alat yang mempermudah banyak perhitungan yang terkait dengan analisis data. Secara spesifik, peralatan dalam modul ini sangat kuat dan bermanfaat dalam perhitungan yang melibatkan matriks dan vektor, yang menjadi objek studi utama dalam mata kuliah MATH2031.

Numpy memang tidak menjadi bawaan standar Python. Oleh sebab itu, untuk dapat menggunakannya, kita perlu melakukan impor terlebih dahulu agar program yang kita tulis memperoleh semua manfaat dari modul Numpy. Dalam prakteknya, proses impor biasanya dilakukan satu kali saja di awal program dengan menulis perintah:

```
import numpy as np
```

Selanjutnya ketika suatu method Numpy yang ingin dipakai di program, kita dapat memanggilnya dengan menuliskan perintah dalam format `np.<nama_method>`

1.1 Mendefinisikan variabel berbentuk vektor

Berikut merupakan contoh cara menyimpan variabel berbentuk vektor dengan perintah `np.array()`. Dalam contoh di bawah ini, kita ingin menyimpan vektor $\begin{bmatrix}2 & 3 & 4\end{bmatrix}$ ke dalam variabel b.

[1]:
```
import numpy as np

b = np.array([2, 3, 4])
print("Vektor b adalah:",b)
```

```
Vektor b adalah: [2 3 4]
```

1.2 Mendefinisikan variabel berbentuk matriks

Terdapat beberapa cara untuk menyimpan variabel dalam bentuk matriks. Kita akan melihat dua metode untuk menyimpan matriks $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\\10 & 11 & 12\end{bmatrix}$ ke dalam variabel A.

[2]:
```
# Cara #1: Menuliskan isi matriks baris demi baris

A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12] ])
print("Matriks A adalah:\n", A)
```

```
Matriks A adalah:
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

[3]:
```
# Cara #2: Menuliskan seluruh isi matriks dalam
# bentuk vektor 1 dimensi kemudian melakukan reshape


# Pertama, A ditulis dalam bentuk vektor 1 dimensi terlebih
# dahulu 1 baris demi 1 baris mulai dari baris teratas
# hingga baris terbawah, dengan perintah di bawah ini.
A = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
print("Matriks A sebelum reshape:", A)

# Kemudian, vektor [1, 2, ... , 12] yang semula
# berdimensi 1 x 12 diubah menjadi matriks 4 x 3
# melalui perintah di bawah ini.
A = A.reshape((4,3))

print("Matriks A setelah reshape:\n",A)
```

```
Matriks A sebelum reshape: [ 1  2  3  4  5  6  7  8  9 10 11 12]
Matriks A setelah reshape:
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

1.3 Beberapa Matriks dan Vektor yang Spesial

Matriks/vektor tertentu dapat dipanggil dengan lebih cepat dengan beberapa fitur Numpy. Berikut beberapa contoh.

1.3.1 Matriks/Vektor Nol

Vektor dengan panjang k yang semua elemennya berisi nol dapat dipanggil dengan perintah `np.zeros(k)` untuk suatu bilangan positif k.

Matriks berukuran $m \times n$ yang semua elemennya berisi nol dapat dipanggil dengan perintah `np.zeros((m,n))` untuk suatu bilangan positif m, n.

[4]:
```
# Memanggil vektor nol dengan panjang 5
b0 = np.zeros(5)
print("Vektor nol dengan panjang 5:", b0)

# Memanggil vektor nol dengan panjang 4 x 3
A0 = np.zeros((4,3))
print("Matriks nol ukuran 4 x 3:\n", A0)
```

```
Vektor nol dengan panjang 5: [0. 0. 0. 0. 0.]
Matriks nol ukuran 4 x 3:
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

1.3.2 Matriks/Vektor Satu

Vektor dengan panjang k yang semua elemennya berisi satu dapat dipanggil dengan perintah `np.ones(k)` untuk suatu bilangan positif k.

Matriks berukuran $m \times n$ yang semua elemennya berisi satu dapat dipanggil dengan perintah `np.ones((m,n))` untuk suatu bilangan positif m, n.

[5]:
```
# Memanggil vektor nol dengan panjang 5
b1 = np.ones(5)
print("Vektor satu dengan panjang 5:",b1)

# Memanggil vektor nol dengan panjang 4 x 3
A1 = np.ones((4,3))
print("Matriks satu ukuran 4 x 3:\n", A1)
```

```
Vektor satu dengan panjang 5: [1. 1. 1. 1. 1.]
Matriks satu ukuran 4 x 3:
 [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

1.3.3 Matriks Identitas

Matriks identitas berukuran $n \times n$, yaitu I_n dapat dipanggil dengan menggunakan perintah `np.eye(n)`

[6]:
```
# Membuat matriks identitas ukuran 4 x 4
I = np.eye(4)
print("Matriks identitas ukuran 4 x 4:\n", I)
```

```
Matriks identitas ukuran 4 x 4:
 [[1. 0. 0. 0.]
```

```
[0. 1. 0. 0.]
[0. 0. 1. 0.]
[0. 0. 0. 1.]]
```

1.3.4 Matriks Diagonal

Misalkan kita ingin membuat matriks diagonal berukuran $n \times n$ yang komponen diagonalnya $d_1, d_2, \ldots d_n$. Maka, matriks yang demikian dapat dipanggil dengan cepat menggunakan Numpy melalui langkah berikut: 1. Definisikan vektor d yang berisi komponen-komponen diagonal tersebut 2. Tulis `D = np.diag(d)` untuk membangun matriks diagonalnya

Di bawah ini contoh cara mendefinisikan variabel berbentuk matriks diagonal 3×3 yang komponen diagonalnya secara berturut-turut adalah $2, -4, 1$.

[7]:
```
# Menuliskan vektor yang berisi komponen diagonal:
x = np.array([2,-4, 1])
print("Komponen diagonal:", x)
# Mengubah vektor menjadi matriks diagonal:
D = np.diag(x)
print("Matriks diagonal:\n", D)
```

```
Komponen diagonal: [ 2 -4  1]
Matriks diagonal:
 [[ 2  0  0]
 [ 0 -4  0]
 [ 0  0  1]]
```

1.3.5 Menggunakan fungsi `np.arange()`

Numpy memiliki sebuah fitur yang sangat bermanfaat untuk menulis vektor atau matriks yang memiliki pola berbentuk barisan aritmatika. Fitur ini memiliki nama `np.arange()` dengan bentuk umum sebagai berikut.

```
np.arange(a, b, s)
```

Perintah seperti di atas akna menghasilkan vektor yang berisi barisan aritmatika yang dimulai dari `a` dan seterusnya hingga kurang dari `b` (batas atas tidak termasuk) dan setiap suku memiliki selisih sebesar `s`, yaitu bilangan-bilangan $a, a+s, a+2s, a+3s, \ldots$ dan seterusnya hingga paling besar b. Sebagai contoh, dengan menulis `np.arange(2, 10, 2)`, Python akan menghasilkan vektor berikut

$$\begin{bmatrix} 2 & 4 & 6 & 8 \end{bmatrix}$$

Perhatikan bahwa elemen 10 tidak dihasilkan pada vektor di atas karena nilai batas atas memang tidak dimasukkan secara definisi Python.

[8]:
```
# Membuat vektor berisi 0, 5, 10, 15, ..., 45
c = np.arange(0, 50, 5)
print("c =", c)

# Membuat matriks 5 x 2 berisi bil kelipatan 5
```

```
C = c.reshape((5,2))
print("C =\n", C)
```

```
c = [ 0  5 10 15 20 25 30 35 40 45]
C =
 [[ 0  5]
 [10 15]
 [20 25]
 [30 35]
 [40 45]]
```

Kita dapat pula membuat barisan aritmatika yang menurun dengan memasukkan nilai negatif pada selisihnya. Perhatikan contoh berikut.

[9]:
```
# Membuat vektor berisi 12, 9, 6, ... , -3
c = np.arange(12, -4, -3)
print("c =", c)

# Membuat versi matriks 3 x 2 nya
C = c.reshape((3,2))
print("C =\n", C)
```

```
c = [12  9  6  3  0 -3]
C =
 [[12  9]
 [ 6  3]
 [ 0 -3]]
```

Latihan 1.
Cetaklah matriks-matriks dengan ketentuan sebagai berikut:

1. Matriks nol berukuran $m \times n$ di mana m adalah tanggal lahir Anda dan n adalah digit terakhir NIM Anda.
2. Matriks identitas berukuran $k \times k$ di mana k merupakan target IP Anda semester ini, dibulatkan ke atas.
3. Matriks diagonal berukuran 3×3 yang komponen diagonalnya berisi 3 bilangan antara 0-100, yaitu secara berturut-turut: target nilai UTS MATH2031 Anda, target nilai UAS MATH2031 Anda, dan target nilai akhir MATH2031 Anda.
4. Matriks berukuran 7×10 yang berisi 70 buah bilangan ganjil yang terurut naik baris demi baris, dimulai dari bilangan ganjil $2t + 1$, di mana t merupakan target berapa banyak mantan Anda sebelum Anda nanti menikah.

2 Operasi Dasar Vektor dan Matriks

Numpy menyediakan berbagai macam fungsi untuk mengoperasikan vektor dan matriks dengan cepat dan mudah. Berikut beberapa penjelasannya

2.1 Penjumlahan Vektor dan Matriks

Syarat: Dimensi vektor/matriks yang dioperasikan harus PERSIS SAMA.

Berikut merupakan contoh perhitungan penjumlahan (atau pengurangan) vektor dan matriks

[10]:
```
# Menghitung vektor x + y dan x - y
x = np.array([2, -1, 5, 7])
y = np.array([1, 1, 7, 4])
print("x =", x)
print("y =", y)

print("x + y =", x + y)
print("x - y =", x - y)

print("\n")

# Menghitung matriks A + B dan A - C
A = x.reshape((2,2))
B = y.reshape((2,2))
print("A =\n", A)
print("B =\n", B)

print("A + B =\n", A + B)
print("A - B =\n", A - B)
```

```
x = [ 2 -1  5  7]
y = [1 1 7 4]
x + y = [ 3  0 12 11]
x - y = [ 1 -2 -2  3]


A =
 [[ 2 -1]
 [ 5  7]]
B =
 [[1 1]
 [7 4]]
A + B =
 [[ 3  0]
 [12 11]]
A - B =
 [[ 1 -2]
 [-2  3]]
```

[11]:
```
# Penjumlahan Matriks akan bernilai ValueError
# jika ukuran matriks yang dijumlahkan tidak sama
x = np.array([2, -1, 5, 7])
print("x =", x)
```

```
A = x.reshape((2,2))
print("A =\n", A)
print("x + A =",x + A)
```

```
x = [ 2 -1  5  7]
A =
 [[ 2 -1]
 [ 5  7]]


      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-11-d963160bfbfe> in <module>
         6 A = x.reshape((2,2))
         7 print("A =\n", A)
    ----> 8 print("x + A =",x + A)


       ValueError: operands could not be broadcast together with shapes (4,)␣
↪(2,2)
```

2.2 Perkalian Skalar dan Kombinasi Linear

Sebuah vektor/matriks dapat dikalikan dengan sebuah angka dengan menggunakan simbol perkalian yang standar, yaitu simbol asterisk (*). Perhatikan contoh berikut.

[12]:
```
# Perkalian skalar pada vektor
x = np.array([2, -1, 5, 7])
print("x =", x)
print("3x =", 3*x)
print("\n")

# Perkalian skalar pada matriks
A = x.reshape((2,2))
print("A =\n", A)
print("-7A =\n", -7*A)
print("\n")

# Kombinasi linear vektor
x = np.array([2, -1, 5, 7])
y = np.array([1, 1, 7, 4])
print("x =", x)
```

```
print("y =", y)
print("3x - 2y =", 3*x - y)
```

```
x = [ 2 -1  5  7]
3x = [ 6 -3 15 21]


A =
[[ 2 -1]
[ 5  7]]
-7A =
[[-14   7]
[-35 -49]]


x = [ 2 -1  5  7]
y = [1 1 7 4]
3x - 2y = [ 5 -4  8 17]
```

2.3 Perkalian Matriks

Terdapat banyak cara/pengertian dalam mengalikan matriks. Dua di antaranya yang paling sering digunakan adalah 1) perkalian *elementwise*, yaitu operasi yang mengalikan elemen demi elemen dengan koordinat yang bersesuaian dari dua buah matriks; dan 2) perkalian matriks dalam pengertian aljabar linear

2.3.1 Perkalian *elementwise*

Dalam operasi perkalian *elementwise*, matriks dikalikan dengan cara mengalikan kedua elemen dari masing2 koordinat matriks. Dalam perkalian jenis ini, disyaratkan bahwa dimensi kedua matriks yang dikalikan harus sama dan jika hal ini tidak dipenuhi, Python akan mengembalikan nilai `ValueError`.

Perintah yang dapat digunakan dalam Python dan Numpy ada beberapa cara, yaitu: * Menggunakan operator perkalian * selayaknya operasi perkalian biasa. * Menggunakan perintah `np.multiply()`

Lebih jelasnya, misalkan kita hendak mengalikan matriks $A, B \in \mathbb{R}^{2\times 2}$ dan misalkan pula elemen dari matriks A dan B adalah sebagai berikut:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Maka, operasi $A * B$ pada Python akan menghasilkan

$$A * B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

Perhatikan contoh di bawah ini.

[13]:
```
# Contoh Perkalian elementwise
A = np.array([[2, -1], [5, 7]])
B = np.array([[1, 1], [7, 4]])
print("Matriks A=\n", A)
print("Matriks B=\n", B)

print("Matriks A*B=\n", A * B)

# Dapat pula menggunakan np. multiply()
print("Dengan np.multiply(), Matriks A*B=\n", np.multiply(A,B))
```

```
Matriks A=
[[ 2 -1]
[ 5  7]]
Matriks B=
[[1 1]
[7 4]]
Matriks A*B=
[[ 2 -1]
[35 28]]
Dengan np.multiply(), Matriks A*B=
[[ 2 -1]
[35 28]]
```

Jika dimensi matriks yang dikalikan tidak sama, operasi perkalian ini akan menghasilkan error seperti contoh di bawah ini.

[14]:
```
C = np.array([[1, 1], [7, 4], [2, 6]])
print("Matriks A=\n", A)
print("Matriks C=\n", C)
A * C
```

```
Matriks A=
[[ 2 -1]
[ 5  7]]
Matriks C=
[[1 1]
[7 4]
[2 6]]


---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-14-7acc20e70a99> in <module>
2 print("Matriks A=\n", A)
3 print("Matriks C=\n", C)
----> 4 A * C
```

```
ValueError: operands could not be broadcast together with shapes (2,2) (3,2)
```

2.3.2 Perkalian Matriks Aljabar Linear

Dalam operasi perkalian dalam pengertian Aljabar Linear, dimensi kedua matriks yang dikalikan harus cocok dalam pengertian sebagai berikut. Misalkan matriks yang dikalikan adalah AB. Maka **banyak kolom** matriks A **harus sama** dengan **banyak baris** matriks B.

Perintah yang dapat digunakan dalam Python dan Numpy ada beberapa cara, yaitu: * Menggunakan operator perkalian @. * Menggunakan perintah `np.dot()`

Lebih jelasnya, misalkan kita hendak mengalikan matriks $A, B \in \mathbb{R}^{2\times 2}$ dan misalkan pula elemen dari matriks A dan B adalah sebagai berikut:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Maka, operasi $A@B$ pada Python akan menghasilkan

$$A@B = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Perhatikan contoh di bawah ini.

[15]:
```python
# Contoh Perkalian Aljabar Linear
A = np.array([[2, -1], [5, 7]])
B = np.array([[1, 1], [7, 4]])
print("Matriks A=\n", A)
print("Matriks B=\n", B)

print("Matriks A@B=\n", A @ B)

# Dapat pula menggunakan np.dot()
print("Dengan np.dot(), Matriks A@B=\n", A.dot(B))
print("Dengan np.dot(), Matriks A@B=\n", np.dot(A,B))
```

```
Matriks A=
[[ 2 -1]
[ 5  7]]
Matriks B=
[[1 1]
[7 4]]
Matriks A@B=
[[-5 -2]
[54 33]]
Dengan np.dot(), Matriks A@B=
```

```
[[-5 -2]
[54 33]]
Dengan np.dot(), Matriks A@B=
[[-5 -2]
[54 33]]
```

Jika dimensi matriks yang dikalikan tidak memenuhi kriteria, operasi perkalian ini akan menghasilkan error seperti contoh di bawah ini.

[16]:
```
C = np.array([[1, 1], [7, 4], [2, 6]])
print("Matriks A=\n", A)
print("Matriks C=\n", C)
A @ C
```

```
Matriks A=
[[ 2 -1]
[ 5  7]]
Matriks C=
[[1 1]
[7 4]
[2 6]]


---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-16-68fcd029fc66> in <module>
2 print("Matriks A=\n", A)
3 print("Matriks C=\n", C)
----> 4 A @ C


ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0,␣
↪with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 3 is different from 2)
```

2.3.3 Transpos Matriks

Numpy juga menyediakan fitur untuk mentranspos matriks secara instan. Berikut merupakan perintah yang dapat dilakukan. Misalkan variabel matriks yang hendak ditranspos adalah `A`, maka transpos dari matriks `A` dapat dihitung dengan perintah: * `A.T`, atau * `A.transpose()`

Perhatikan contoh di bawah ini.

[17]:
```
# Operasi Transpos Matriks
A = np.array([[2, -1], [5, 7]])
print("Matriks A=\n", A)

print("Transpos matriks A=\n", A.T)
```

```
print("Transpos matriks A=\n", A.transpose())
```

```
Matriks A=
[[ 2 -1]
[ 5  7]]
Transpos matriks A=
[[ 2  5]
[-1  7]]
Transpos matriks A=
[[ 2  5]
[-1  7]]
```

2.3.4 Menggunakan operasi matriks untuk menyusun matriks yang lebih kompleks

Contoh 1. Untuk membuat matriks A seperti di bawah ini

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix},$$

kita dapat memanipulasi matriks satu dengan perkalian skalar, yaitu

$$A = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} = 2 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Secara Python, kita dapat melakukan langkah seperti di bawah ini.

[18]:
```
satu = np.ones((2,3))
print("Awalnya definisikan matriks satu:\n", satu)
A = 2 * satu
print("Kemudian kalikan dengan skalar 2 untuk mendapatkan matriks A:\n", A)
```

```
Awalnya definisikan matriks satu:
[[1. 1. 1.]
[1. 1. 1.]]
Kemudian kalikan dengan skalar 2 untuk mendapatkan matriks A:
[[2. 2. 2.]
[2. 2. 2.]]
```

Contoh 2. Misalkan kita hendak membuat matriks B seperti di bawah ini

$$B = \begin{bmatrix} 1 & 4 & 5 \\ 1 & 4 & 5 \\ 1 & 4 & 5 \end{bmatrix}.$$

Perhatikan bahwa matriks B dapat diperoleh dari matriks satu berukuran 3×3 dengan cara melakukan perkalian skalar kolom 1 dengan 1, kolom 2 dengan 4 dan kolom 3 dengan 5. Hal ini dapat diperoleh dengan mengalikan matriks satu tersebut dengan matriks diagonal dengan komponen diagonal $[1, 4, 5]$, yaitu

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 5 \\ 1 & 4 & 5 \\ 1 & 4 & 5 \end{bmatrix} = B.$$

Secara Python, kita dapat melakukan langkah seperti di bawah ini.

[19]:
```
satu = np.ones((3,3))
print("Awalnya definisikan matriks satu:\n", satu)
D = np.diag([1, 4, 5])
print("Kemudian definisikan matriks diagonal:\n", D)
B = satu @ D
print("Kalikan matriks satu dengan matriks diagonal untuk mendapatkan matriks B:
 ↪\n", B)
```

```
Awalnya definisikan matriks satu:
[[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]]
Kemudian definisikan matriks diagonal:
[[1 0 0]
[0 4 0]
[0 0 5]]
Kalikan matriks satu dengan matriks diagonal untuk mendapatkan matriks B:
[[1. 4. 5.]
[1. 4. 5.]
[1. 4. 5.]]
```

Latihan 2.
Misalkan

$$A = \begin{bmatrix} 1 & 5 & 2 & 9 \\ 2 & 7 & -1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 8 \\ -3 & 4 \\ 1 & 1 \\ 7 & -2 \end{bmatrix}.$$

Dengan Python, hitunglah nilai:

1. $20A - 7B^T$
2. $AA^T - AB + B^TB$
3. $(BB^T)^5 - 10B(AA^T)^5B^T + I$

Misalkan pula

$$C = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 & 1 \\ 3 & 3 & 3 & \dots & 3 & 3 \\ 5 & 5 & 5 & \dots & 5 & 5 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 99 & 99 & 99 & \dots & 99 & 99 \end{bmatrix}, \qquad D = \begin{bmatrix} 100 & 98 & 96 & \dots & 4 & 2 \\ 100 & 98 & 96 & \dots & 4 & 2 \\ 100 & 98 & 96 & \dots & 4 & 2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 100 & 98 & 96 & \dots & 4 & 2 \end{bmatrix}.$$

Dengan Python, hitunglah nilai:

4. CD
5. $C^3/100 + D^2/25$

Soal Tantangan (Perlu Keterampilan Pemrograman).
Dapatkah Anda menulis algoritma operasi perkalian matriks secara Aljabar Linear tanpa menggunakan fitur dari Numpy?

```
Input  : 1) Bilangan bulat k, m, dan n
         2) Matriks A berukuran k x m
         3) Matriks B berukuran m x n

Output : Matriks berukuran k x n yang adalah perkalian AB
```

Petunjuk: gunakan perulangan bersarang (nested loop)