

Nama : Iyan Zuli Armanda

NIM : 23051204165

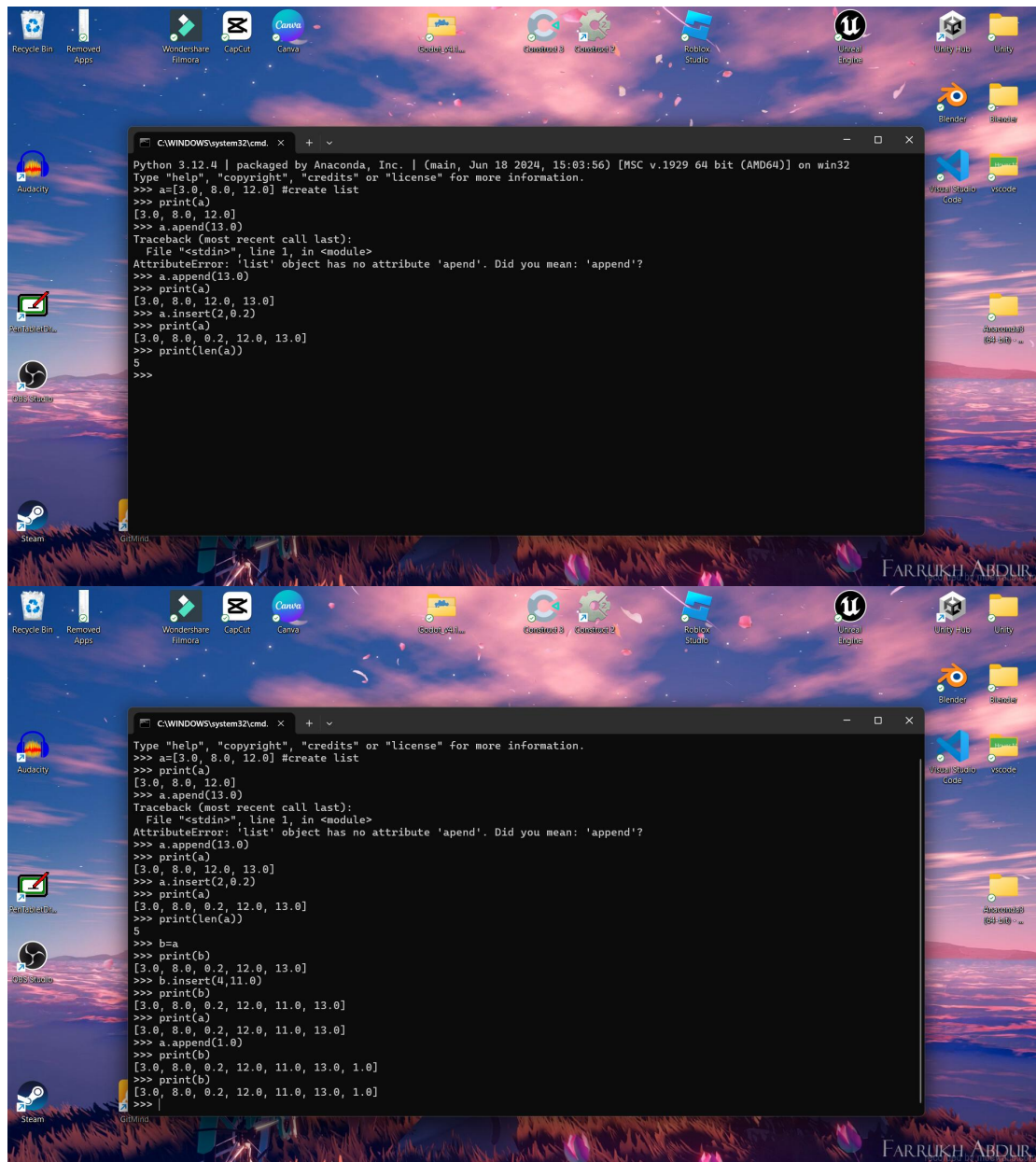
Kelas : TI E 23

Berikut adalah hasil dari praktek menggunakan anaconda dengan bahasa python di cmd terhadap operasi matematika :



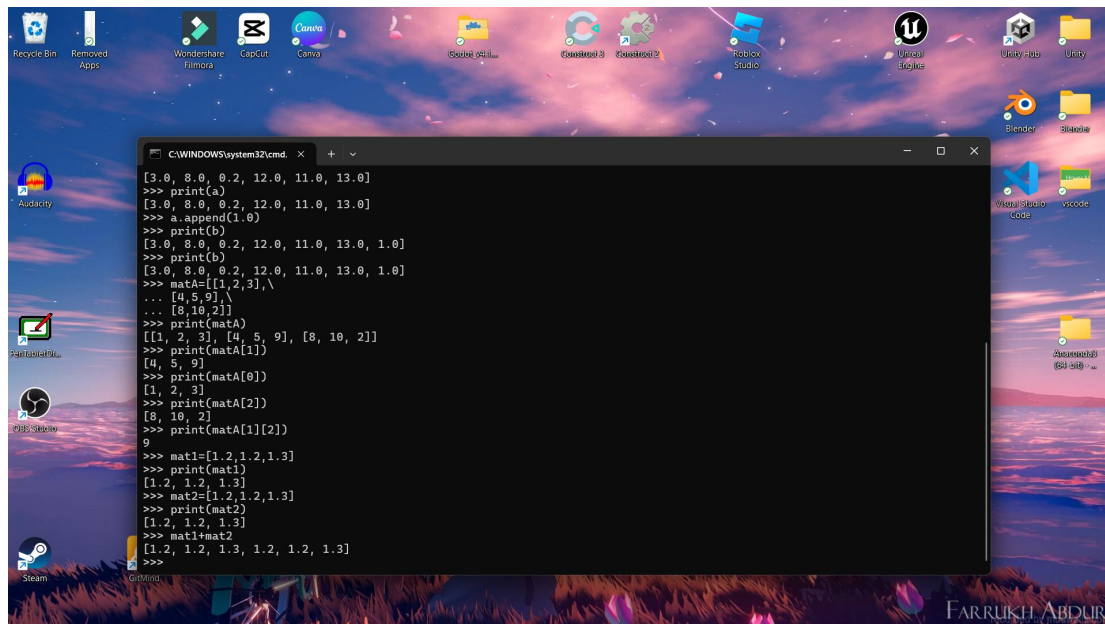
```
Python 3.12.5 | packaged by Anaconda, Inc. | (main, Sep 12 2024, 18:18:29) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import math
>>> a=9
>>> sqrt(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>> import math
>>> a=9
>>> math.sqrt(a)
3.0
>>> import math as matematika
>>> akar=matematika.sqrt(9)
>>> print(akar)
3.0
>>> pembulatanAtas=matematika.ceil(6.5)
>>> print(pembulatanAtas)
7
>>> power=matematika.pow(2,3)
>>> print(power)
8.0
>>>
```

- Math adalah semacam syntax untuk mendefinisikan matematika, dan definisi itu dipanggil ke dalam cmd dengan bahasa python
- sqrt adalah akar dari. Akar dari 9 adalah 3
- ceil adalah pembulatan ke atas. Pembulatan ke atas dari 6.5 dan 6.1 adalah 7
- power adalah pangkat. Pow(2,3) artinya 2^3 , jawabannya adalah 8



```
C:\WINDOWS\system32\cmd. x + v
Python 3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a=[3.0, 8.0, 12.0] #create list
>>> print(a)
[3.0, 8.0, 12.0]
>>> a.append(13.0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute 'append'. Did you mean: 'append'?
>>> print(a)
[3.0, 8.0, 12.0, 13.0]
>>> a.insert(2,0.2)
>>> print(a)
[3.0, 8.0, 0.2, 12.0, 13.0]
>>> print(len(a))
5
>>>
>>>
>>> b=a
>>> print(b)
[3.0, 8.0, 0.2, 12.0, 13.0]
>>> b.insert(4,11.0)
>>> print(b)
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0]
>>> print(a)
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0]
>>> a.append(1.0)
>>> print(b)
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0, 1.0]
>>> print(b)
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0, 1.0]
>>>
>>> |
```

Arti perintah `b=a` dalam program diatas ialah definisi bahwa nilai `b` akan selalu sama dengan `a`. Jadi ketika nilai `a` dirubah, maka nilai `b` dirubah dan sebaliknya, jika nilai `b` berubah maka nilai `a` berubah pula.



```
C:\WINDOWS\system32\cmd.  
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0]  
>>> print(a)  
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0]  
>>> a.append(1.0)  
>>> print(a)  
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0, 1.0]  
>>> print(b)  
[3.0, 8.0, 0.2, 12.0, 11.0, 13.0, 1.0]  
>>> matA=[[1,2,3],\  
... [4,5,9],\  
... [8,10,2]]  
>>> print(matA)  
[[1, 2, 3], [4, 5, 9], [8, 10, 2]]  
>>> print(matA[1])  
[4, 5, 9]  
>>> print(matA[0])  
[1, 2, 3]  
>>> print(matA[2])  
[8, 10, 2]  
>>> print(matA[1][2])  
9  
>>> mat1=[1.2,1.2,1.3]  
>>> print(mat1)  
[1.2, 1.2, 1.3]  
>>> mat2=[1.2,1.2,1.3]  
>>> print(mat2)  
[1.2, 1.2, 1.3]  
>>> mat1+mat2  
[1.2, 1.2, 1.3, 1.2, 1.2, 1.3]  
>>>
```

Dalam program itu, telah dibuat matriks bernama A [1, 2, 3]
[4, 5, 9]
[8, 10, 2]

Jika menginput matA[1], maka hanya muncul baris kedua saja berupa [4, 5, 9].

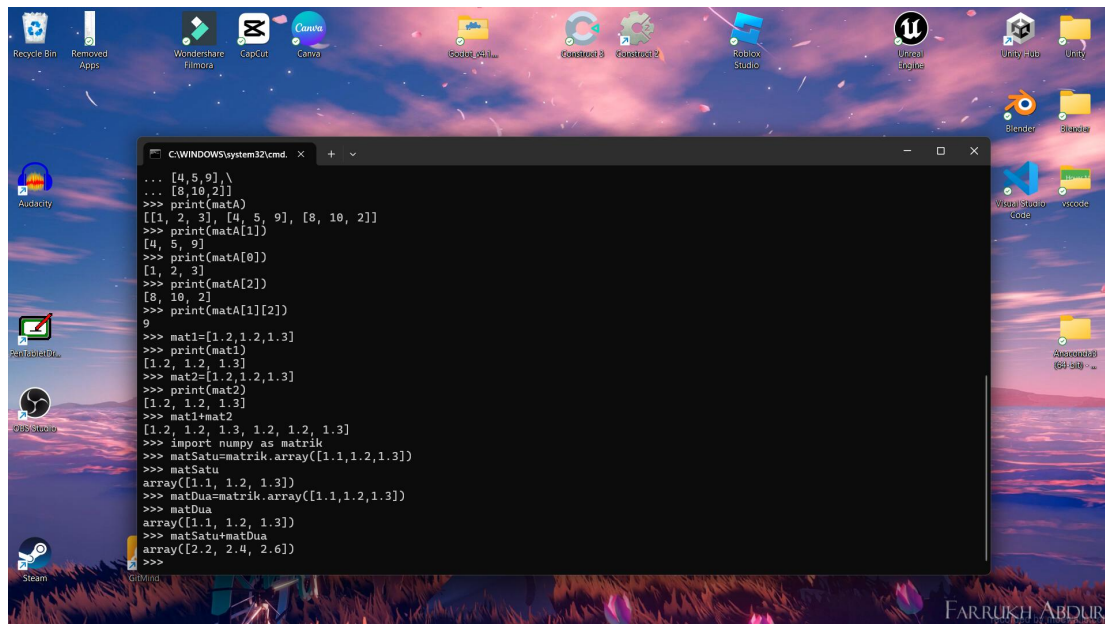
Jika menginput matA[0], maka hanya muncul baris pertama saja berupa [1, 2, 3].

Jika menginput matA[2], maka hanya muncul baris ketiga saja berupa [8, 10, 2].

Jika menginput matA[1][2], maka hanya muncul 1 angka dari baris kedua dan kolom ketiga saja, yaitu 9.

Terdapat pendefinisian matriks 1 [1.2, 1.2, 1.3] dan matriks 2 [1.2, 1.2, 1.3].

Input mat1+mat2 akan menghasilkan penggabungan kedua matriks tersebut,
[1.2, 1.2, 1.3, 1.2, 1.2, 1.3]



```
C:\WINDOWS\system32\cmd. x + -
... [4, 5, 9], \
... [8, 10, 2]]
>>> print(matA)
[[1, 2, 3], [4, 5, 9], [8, 10, 2]]
>>> print(matA[1])
[4, 5, 9]
>>> print(matA[0])
[1, 2, 3]
>>> print(matA[2])
[8, 10, 2]
>>> print(matA[1][2])
9
>>> mat1=[1.2,1.2,1.3]
>>> print(mat1)
[1.2, 1.2, 1.3]
>>> mat2=[1.2,1.2,1.3]
>>> print(mat2)
[1.2, 1.2, 1.3]
>>> mat1+mat2
[1.2, 1.2, 1.3, 1.2, 1.2, 1.3]
>>> import numpy as matrik
>>> matSatu=matrik.array([1.1,1.2,1.3])
>>> matSatu
array([1.1, 1.2, 1.3])
>>> matDua=matrik.array([1.1,1.2,1.3])
>>> matDua
array([1.1, 1.2, 1.3])
>>> matSatu+matDua
array([2.2, 2.4, 2.6])
>>>
```

Implementasi numpy sebagai matriks ialah membuat input $mat+mat$ menjadi ditambah karena tipe data berbentuk array, bukan digabung seperti yang sebelumnya.

Contohnya $matSatu$ $[1.1, 1.2, 1.3]$ dan $matDua$ $[1.1, 1.2, 1.3]$

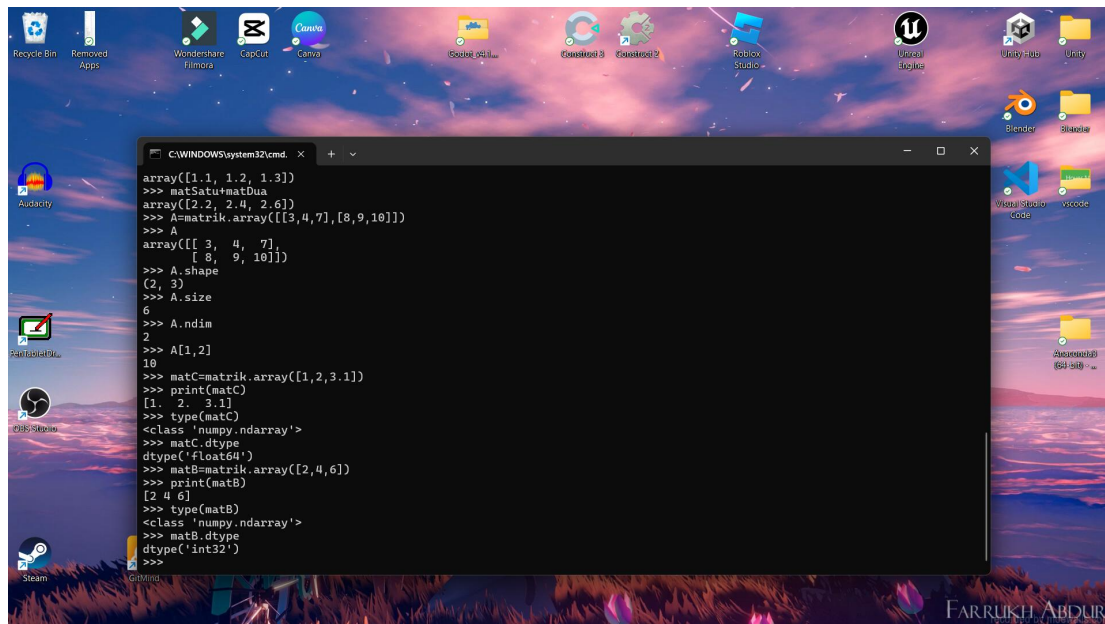
$matSatu+matDua = [2.2, 2.4, 2.6]$.

Selain itu, numpy juga memiliki beberapa fungsi lain sebagai berikut (contoh cmd ada di bawah):

$A = [3, 4, 7]$

$[8, 9, 10]$

- shape, dapat menyebutkan bentuk dari matriks. Contohnya matriks A berbentuk (2,3), artinya terdapat 2 baris dan 3 kolom
- size, dapat menyebutkan total elemen dari matriks tersebut. Contohnya matriks A memiliki total elemen 6, yaitu 3, 4, 7, 8, 9, dan 10.
- ndim, dapat menentukan dimensi dari matriks. Contohnya matriks A memiliki 2 dimensi, baris dan kolom

A screenshot of a Windows desktop with a colorful, abstract background. The desktop is cluttered with various application icons, including Recycle Bin, Removed Apps, Wondershare Filmora, CapCut, Canva, Sketch, Unreal Engine 3, Unreal Engine 2, Roblox Studio, Unreal Engine, Unity Hub, Unity, Blender, Sketcher, Visual Studio Code, and VS Code. A terminal window titled 'C:\WINDOWS\system32\cmd.' is open in the center, displaying a series of NumPy array operations and their results. The operations include creating arrays, adding them, and checking their shapes, sizes, and data types. The data types shown are 'float64' for matC and 'int32' for matB.

```
C:\WINDOWS\system32\cmd. x + v
array([1.1, 1.2, 1.3])
>>> matSatu+matDua
array([2.2, 2.4, 2.6])
>>> A=matrik.array([[3,4,7],[8,9,10]])
>>> A
array([[ 3,  4,  7],
       [ 8,  9, 10]])
>>> A.shape
(2, 3)
>>> A.size
6
>>> A.ndim
2
>>> A[1,2]
10
>>> matC=matrik.array([1,2,3.1])
>>> print(matC)
[1.  2.  3.1]
>>> type(matC)
<class 'numpy.ndarray'>
>>> matC.dtype
dtype('float64')
>>> matB=matrik.array([2,4,6])
>>> print(matB)
[2 4 6]
>>> type(matB)
<class 'numpy.ndarray'>
>>> matB.dtype
dtype('int32')
>>>
```

Yang dapat disimpulkan dalam penentuan tipe data dalam komputasi ialah untuk efisiensi memori dan kecepatan proses. Sebagai contoh, matC bertipe data float 64 bit (terdapat bilangan desimal seperti 3.1) yang cocok untuk menyimpan data dengan presisi tinggi walau kecepatan proses berkurang. Berbeda dengan matB yang bertipe data int 32 bit (bilangan bulat tanpa koma), lebih cepat karena data lebih simpel walau kurang akurat. Baik matC maupun matB, memiliki tipe numpy, yaitu data nya digambarkan ke dalam array agar perhitungan lebih kompleks.

Dalam permasalahan komputasi, seorang engineer harus bisa memperhitungkan efisiensi dan efektivitas proses seperti akurasi data dan kecepatan proses. Semua itu digantungkan berdasarkan kebutuhan data, apakah memerlukan akurasi tinggi atau tidak dan tentunya memerlukan kecepatan yang wajar.