



Deep Learning Toolkit (*Numpy*)

Rowel Atienza, PhD

University of the Philippines

github.com/roatienza

2022

Outline

- Environment, Code Editor
- Python
- **Tensor libraries – numpy, einsum, einops**
- PyTorch, Timm
- Huggingface (HF), Gradio, Streamlit
- HF Accelerator, GitHub
- Machines – Colab, DeepNote, Kaggle, SageMaker
- Other tools

Numpy

<https://numpy.org/>

List vs Tuple or

`a=[1, 2.2, "the"]` vs `a=(1, 2.2, "the")`

	List	Tuple
Mutable	Yes	No
Supported	<code>index, count</code>	<code>index, count</code>
Supported	<code>insert, append, pop, clear, remove, reverse</code>	
Use	Elements might change	Fixed elements

Numpy - Basics

Create an array

```
import numpy as np
```

```
a = np.array([[1,2,3], [4,5,6]])
```

Data type: dtype('int64')

```
a.dtype
```

Shape: (2, 3)

```
a.shape
```

Number of dimensions: 2

```
a.ndim
```

Numpy - Basics

Add a constant

`2 + a`

Add 2 arrays

`b = np.ones(a.shape)`

`a+b`

Multiply 2 arrays

`a*b`

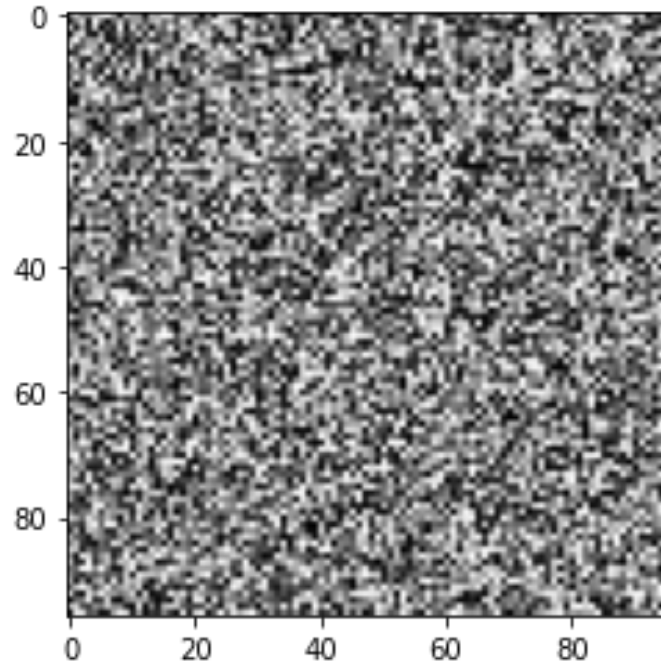
Matrix multiply 2 arrays

`np.matmul(a, np.transpose(b))`

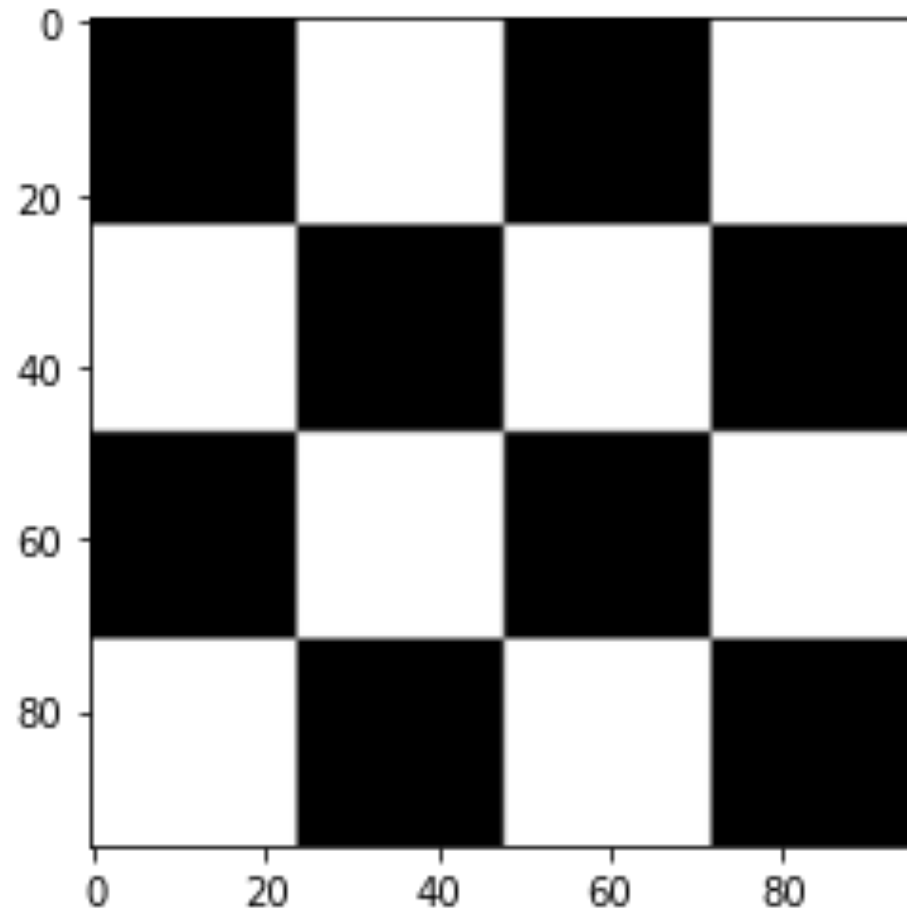
`a@np.transpose(b)`

Numpy for Data

```
img = np.random.randint(0,255,size=(96,96),dtype=np.uint8)  
plt.imshow(img, cmap='gray', vmin=0, vmax=255)  
plt.show()
```



Chessboard Pattern

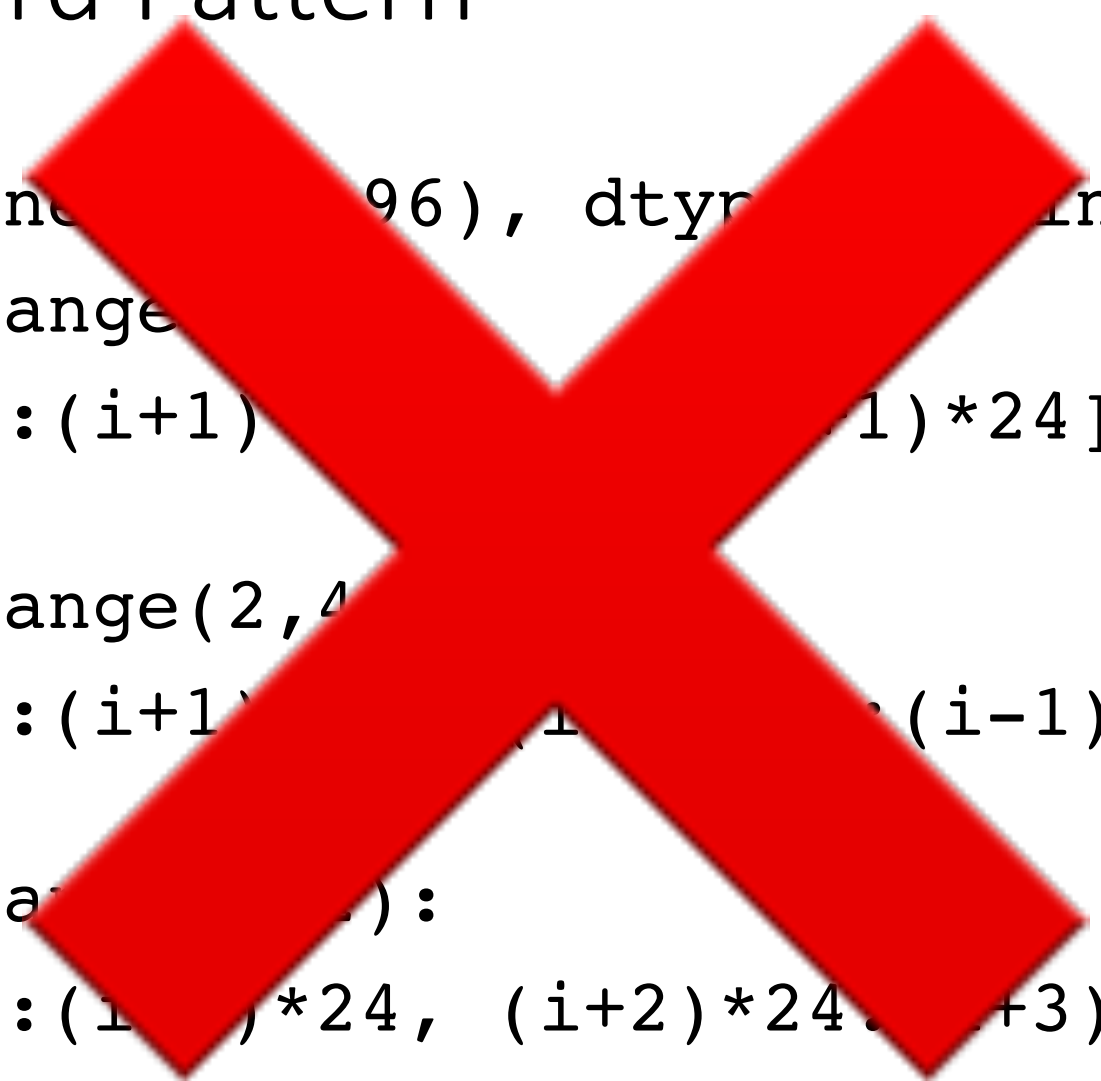


Chessboard Pattern

```
img = np.zeros((4,4,3), dtype=np.uint8)*255
for i in range(0,2):
    img[i*24:(i+1)*24] = 0

for i in range(2,4):
    img[i*24:(i+1)*24, (i-1)*24] = 0

for i in range(0,2):
    img[i*24:(i+2)*24, (i+3)*24] = 0
```



Chessboard Pattern

```
def chessboard(shape):  
    return np.indices(shape).sum(axis=0) % 2  
  
img = chessboard((4,4))*255  
  
img = np.repeat(img, (24), axis=0)  
img = np.repeat(img, (24), axis=1)
```

```
np.indices( ( 4 , 4 ) )
```

```
[[[ 0  0  0  0]  
   [ 1  1  1  1]  
   [ 2  2  2  2]  
   [ 3  3  3  3]]]
```

```
[[[ 0  1  2  3]  
   [ 0  1  2  3]  
   [ 0  1  2  3]  
   [ 0  1  2  3]]]
```

```
np.indices( ( 4 , 4 ) ) .sum( axis=0 )
```

```
[[ 0  1  2  3]
 [ 1  2  3  4]
 [ 2  3  4  5]
 [ 3  4  5  6]]
```

```
np.indices( ( 4 , 4 ) ) .sum( axis=0 ) % 2
```

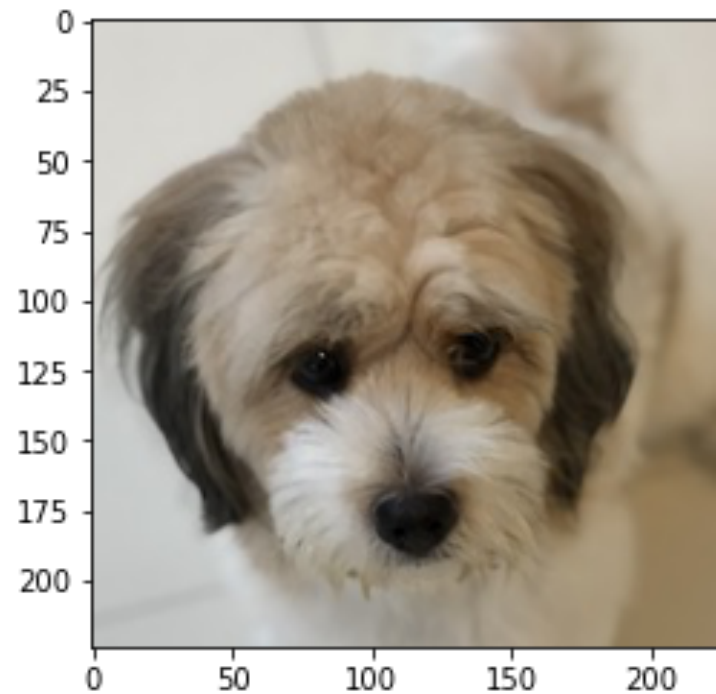
```
[[ 0  1  0  1]
 [ 1  0  1  0]
 [ 0  1  0  1]
 [ 1  0  1  0]]
```

Exercise:

Without using loops, find another algorithm that can generate this pattern.

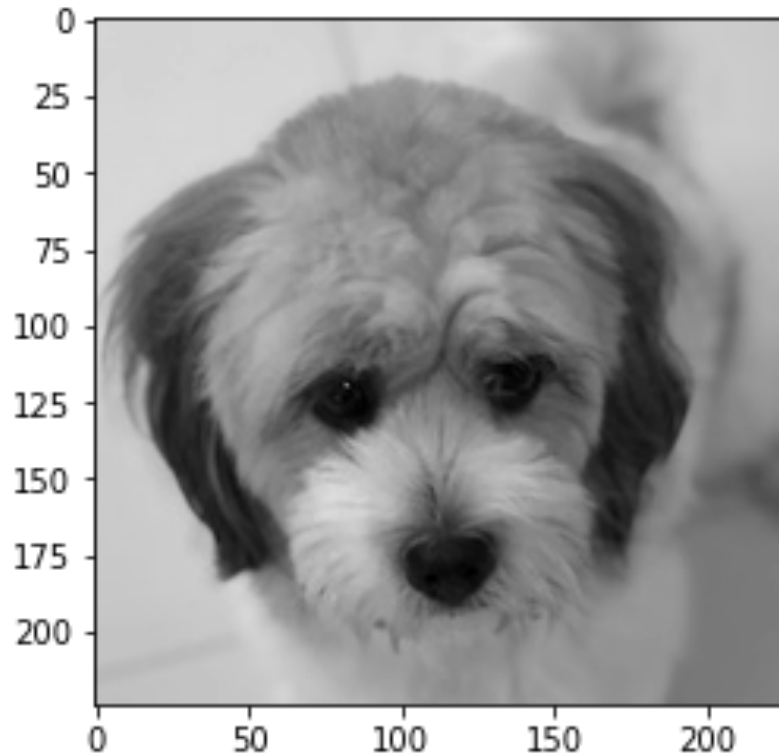
Loading an image

```
from matplotlib import image  
img = image.imread("aki_dog.jpg")
```



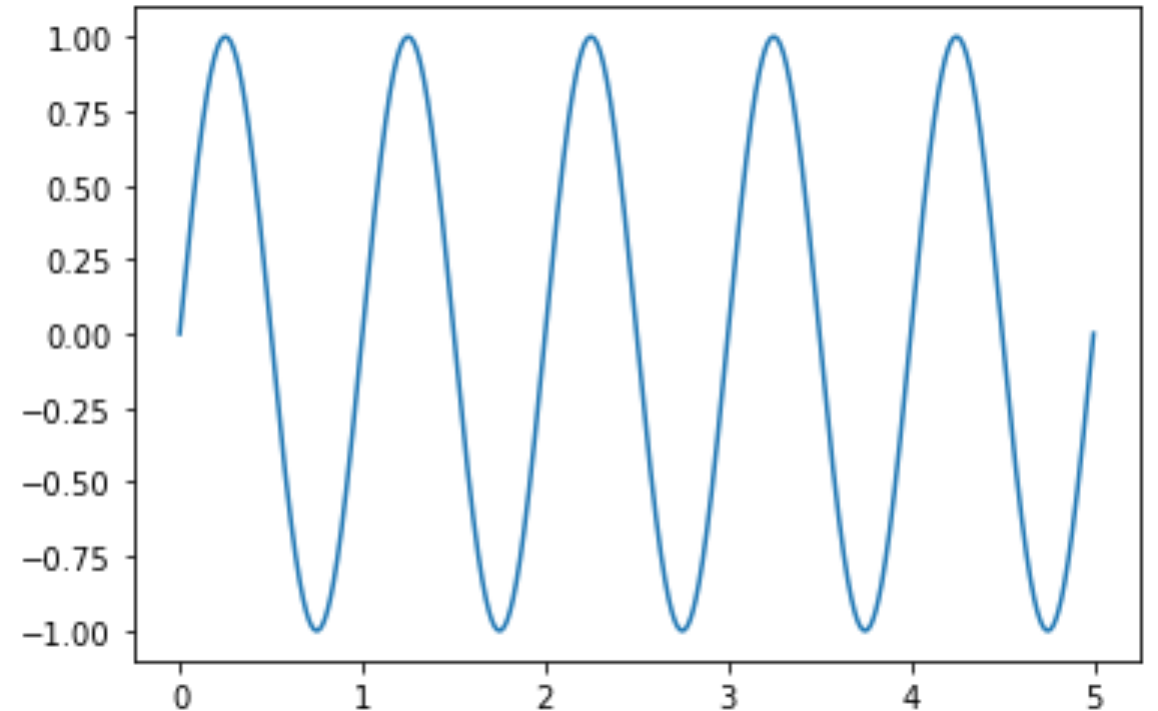
RGB to Grayscale

```
img = np.mean(img, axis=-1)
```



Synthetic Audio Waveform

```
samples_per_sec = 22050  
freq = 1  
n_points = samples_per_sec*5  
t = np.linspace(0,5,n_points)  
data = np.sin(2*np.pi*freq*t)
```



Limitations of Numpy

- Not designed for GPU execution
 - Alternative: `cupy`
- Different methods/APIs for different tensor operations
- Many steps for complex linear algebra operations
 - Alternative: `einsum` and `einops`

End

https://github.com/roatienza/Deep-Learning-Experiments/blob/master/versions/2022/tools/python/np_demo.ipynb