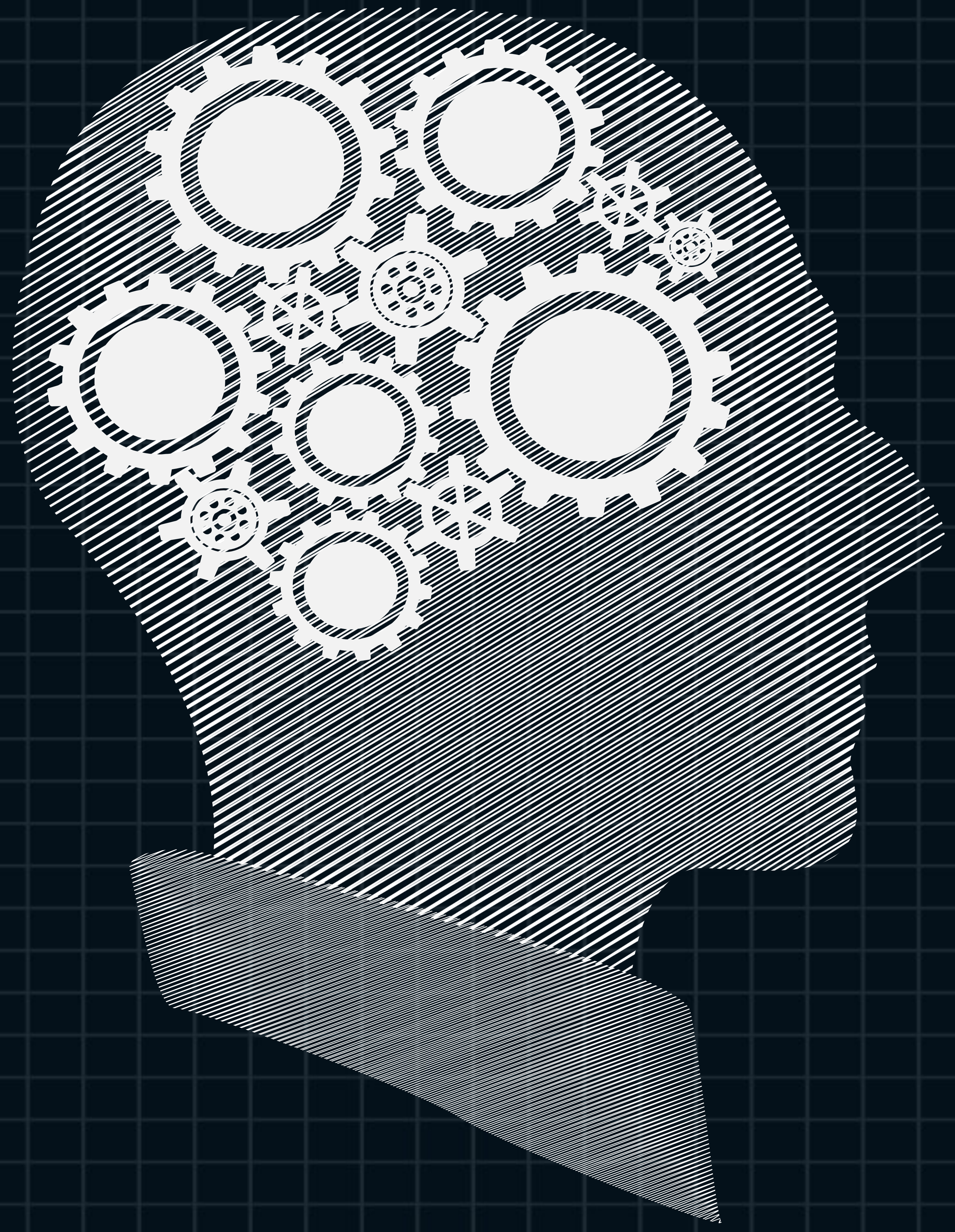
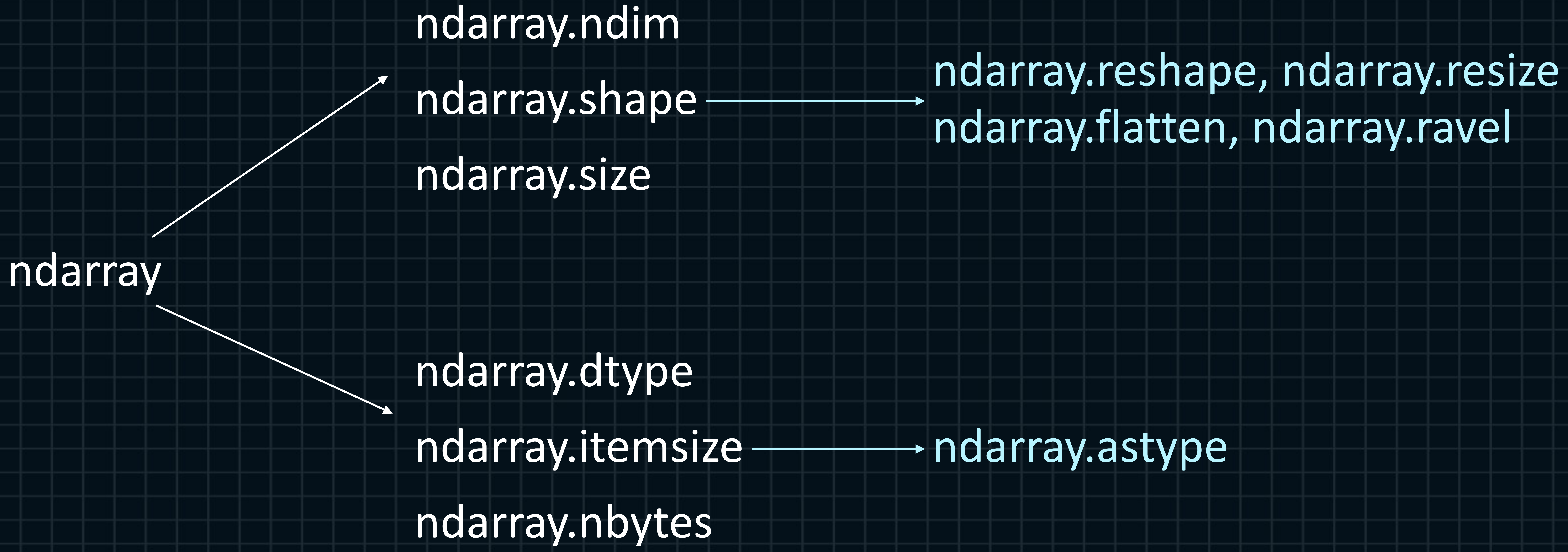


NumPy Master Class

Lecture.4
Meta-data of ndarrays

- Meta-data of ndarrays



ndarray

ndarray.ndim

ndarray.shape

ndarray.size

ndarray.reshape, ndarray.resize

ndarray.flatten, ndarray.ravel

ndarray.dtype

ndarray.itemsize

ndarray.nbytes

ndarray.astype

Lecture.4
Meta-data of ndarrays

- ndim, shape, and size

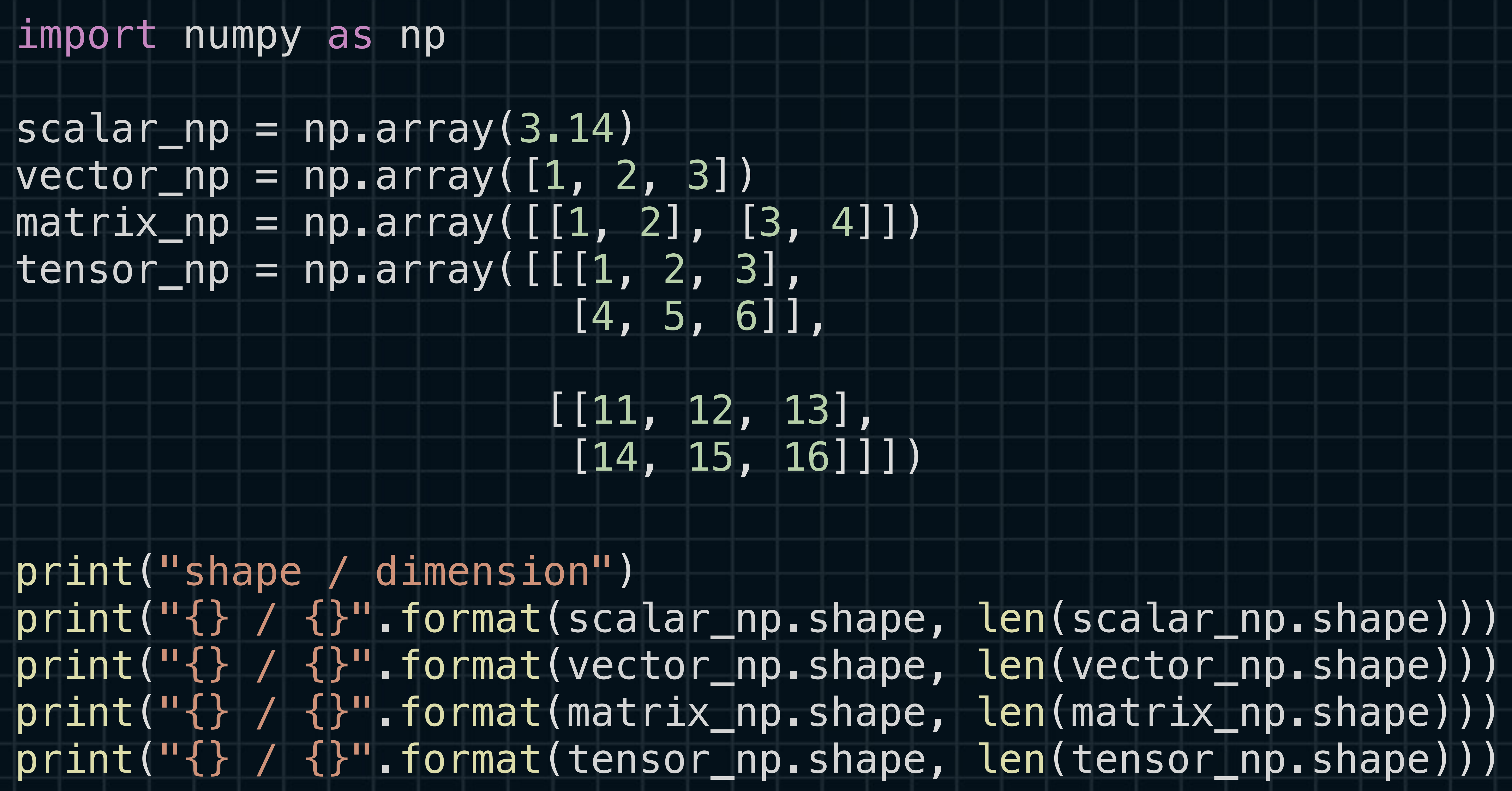
ndarray.ndim

```
import numpy as np

scalar_np = np.array(3.14)
vector_np = np.array([1, 2, 3])
matrix_np = np.array([[1, 2], [3, 4]])
tensor_np = np.array([[[1, 2, 3],
                       [4, 5, 6]],

                      [[11, 12, 13],
                       [14, 15, 16]]])


print(scalar_np.ndim)  0
print(vector_np.ndim)  1
print(matrix_np.ndim)  2
print(tensor_np.ndim)  3
```

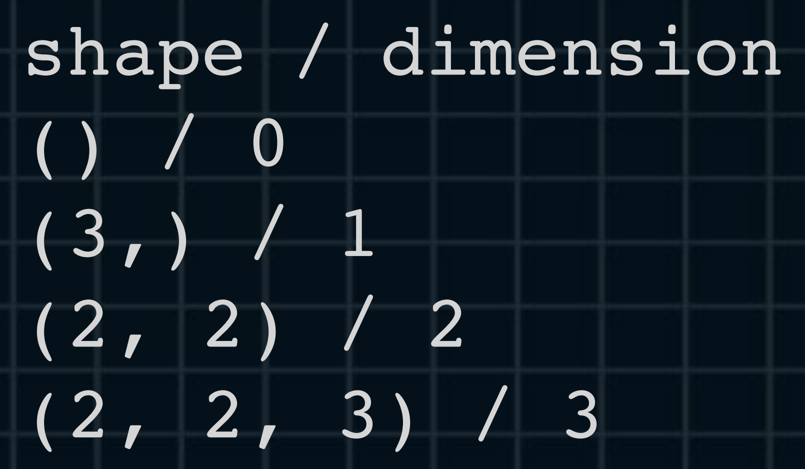
Lecture.4
Meta-data of ndarrays

- ndim, shape, and size

ndarray.shape



```
import numpy as np

scalar_np = np.array(3.14)
vector_np = np.array([1, 2, 3])
matrix_np = np.array([[1, 2], [3, 4]])
tensor_np = np.array([[[1, 2, 3],
                       [4, 5, 6]],

                      [[11, 12, 13],
                       [14, 15, 16]]])


print("shape / dimension")
print("{} / {}".format(scalar_np.shape, len(scalar_np.shape)))
print("{} / {}".format(vector_np.shape, len(vector_np.shape)))
print("{} / {}".format(matrix_np.shape, len(matrix_np.shape)))
print("{} / {}".format(tensor_np.shape, len(tensor_np.shape)))
```



```
shape / dimension
() / 0
(3,) / 1
(2, 2) / 2
(2, 2, 3) / 3
```

Lecture.4
Meta-data of ndarrays - ndim, shape, and size

ndarray.shape

```python
import numpy as np

a = np.array([1, 2, 3])
b = np.array([[1, 2, 3]])
c = np.array([[1], [2], [3]])

print(f"a: {a.shape}\n{a}\n")
print(f"b: {b.shape}\n{b}\n")
print(f"c: {c.shape}\n{c}\n")
```

```
a: (3,)
[1 2 3]

b: (1, 3)
[[1 2 3]]

c: (3, 1)
[[1]
 [2]
 [3]]
```

Lecture.4
Meta-data of ndarrays

- ndim, shape, and size

ndarray.size

```python
import numpy as np

M = np.ones(shape=(10, ))
N = np.ones(shape=(3, 4))
O = np.ones(shape=(3, 4, 5))
P = np.ones(shape=(2, 3, 4, 5, 6))


print("Size of M:", M.size)
print("Size of N:", N.size)
print("Size of O:", O.size)
print("Size of P:", P.size)
```

```
Size of M: 10
Size of N: 12
Size of O: 60
Size of P: 720
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

Data types in NumPy

np.int	np.uint	np.float	np.complex
np.int8	np.uint8		
np.int16	np.uint16		
np.int32	np.uint32	np.float32	
np.int64	np.uint64	np.float64	np.complex64
			np.complex128

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.dtype

```
import numpy as np

M = np.arange(100)
N = np.full(fill_value=3.14, shape=(2, 3))


print(M.dtype)  int64
print(N.dtype)  float64
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.dtype

```
import numpy as np

int_np = np.array([1, 2, 3])
float_np = np.array([1., 2., 3.])


print(int_np.dtype)    int64
print(float_np.dtype)  float64
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.dtype

```python
import numpy as np

int8_np = np.array([1, 2, 3], dtype=np.int8)
int16_np = np.array([1, 2, 3], dtype=np.int16)
int32_np = np.array([1, 2, 3], dtype=np.int32)
int64_np = np.array([1, 2, 3], dtype=np.int64)

uint8_np = np.array([1, 2, 3], dtype=np.uint8)
uint16_np = np.array([1, 2, 3], dtype=np.uint16)
uint32_np = np.array([1, 2, 3], dtype=np.uint32)
uint64_np = np.array([1, 2, 3], dtype=np.uint64)

float32_np = np.array([1, 2, 3], dtype=np.float32)
float64_np = np.array([1, 2, 3], dtype=np.float64)

print("Interger: {}/{}/{}/{}".format(int8_np.dtype, int16_np.dtype,
                                     int32_np.dtype, int64_np.dtype))
print("Unsigned Integer: {}/{}/{}/{}".format(uint8_np.dtype, uint16_np.dtype,
                                             uint32_np.dtype, uint64_np.dtype))
print("Floating Point: {}/{}".format(float32_np.dtype, float64_np.dtype))
```

```
Interger: int8/int16/int32/int64
Unsigned Integer: uint8/uint16/uint32/uint64
Floating Point: float32/float64
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.dtype

```
import numpy as np

int8_np = np.array([1.5, 2.5, 3.5], dtype=np.int8)
uint8_np = np.array([1.5, 2.5, 3.5], dtype=np.uint8)


print(int8_np)    [1 2 3]
print(uint8_np)   [1 2 3]
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.dtype

```python
import numpy as np

M = np.ones(shape=(2, 3), dtype=np.float32)
N = np.zeros_like(M, dtype=np.float64)

print("{}/{}".format(M.dtype, N.dtype))
print(M)
print(N)
```

```
float32/float64
[[1. 1. 1.]
 [1. 1. 1.]]
[[0. 0. 0.]
 [0. 0. 0.]]
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.itemsize

```
import numpy as np

int8_np = np.array([1, 2, 3], dtype=np.int8)
int16_np = np.array([1, 2, 3], dtype=np.int16)
int32_np = np.array([1, 2, 3], dtype=np.int32)
int64_np = np.array([1, 2, 3], dtype=np.int64)

uint8_np = np.array([1, 2, 3], dtype=np.uint8)
uint16_np = np.array([1, 2, 3], dtype=np.uint16)
uint32_np = np.array([1, 2, 3], dtype=np.uint32)
uint64_np = np.array([1, 2, 3], dtype=np.uint64)

float32_np = np.array([1, 2, 3], dtype=np.float32)
float64_np = np.array([1, 2, 3], dtype=np.float64)


print("int8_np: {}/{}B".format(int8_np.dtype, int8_np.itemsize))
print("int16_np: {}/{}B".format(int16_np.dtype, int16_np.itemsize))
print("int32_np: {}/{}B".format(int32_np.dtype, int32_np.itemsize))
print("int64_np: {}/{}B\n".format(int64_np.dtype, int64_np.itemsize))

print("uint8_np: {}/{}B".format(uint8_np.dtype, uint8_np.itemsize))
print("uint16_np: {}/{}B".format(uint16_np.dtype, uint16_np.itemsize))
print("uint32_np: {}/{}B".format(uint32_np.dtype, uint32_np.itemsize))
print("uint64_np: {}/{}B\n".format(uint64_np.dtype, uint64_np.itemsize))

print("float32_np: {}/{}B".format(float32_np.dtype, float32_np.itemsize))
print("float64_np: {}/{}B".format(float64_np.dtype, float64_np.itemsize))
```

```
int8_np: int8/1B
int16_np: int16/2B
int32_np: int32/4B
int64_np: int64/8B

uint8_np: uint8/1B
uint16_np: uint16/2B
uint32_np: uint32/4B
uint64_np: uint64/8B

float32_np: float32/4B
float64_np: float64/8B
```

Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.itemsize

```
import numpy as np

normal = np.random.normal(size=(50, 50, 32, 5))

print("size: ", normal.size)
print("dtype/itemsize: {}/{}\n".format(normal.dtype, normal.itemsize))
```

```
size:  400000
dtype/itemsize: float64/8
```

```
m_cap = normal.size * normal.itemsize

print("Memory capacity in B: {}B".format(m_cap))
print("Memory capacity in KB: {}KB".format(m_cap/1024))
print("Memory capacity in MB: {}MB".format(m_cap/1024**2))
```

```
Memory capacity in B: 3200000B
Memory capacity in KB: 3125.0KB
Memory capacity in MB: 3.0517578125MB
```

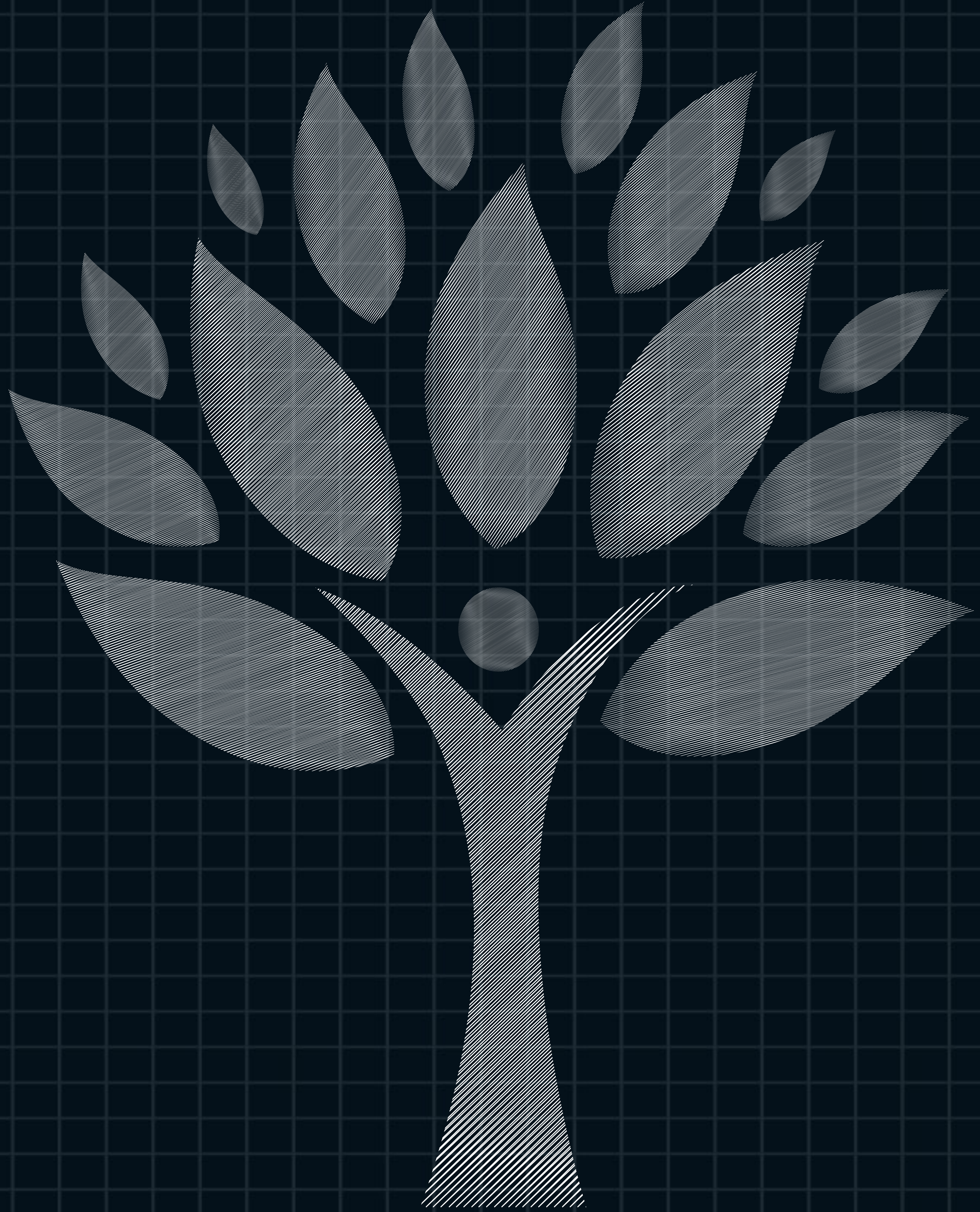
Lecture.4
Meta-data of ndarrays

- dtype, itemsize and nbytes

ndarray.nbytes

```python
import numpy as np

normal = np.random.normal(size=(50, 50, 32, 5))


m_cap = normal.size * normal.itemsize
print("{}B/{}B".format(m_cap, normal.nbytes))
```

```
    3200000B/3200000B
```



NumPy Master Class

Lecture.4
Meta-data of ndarrays