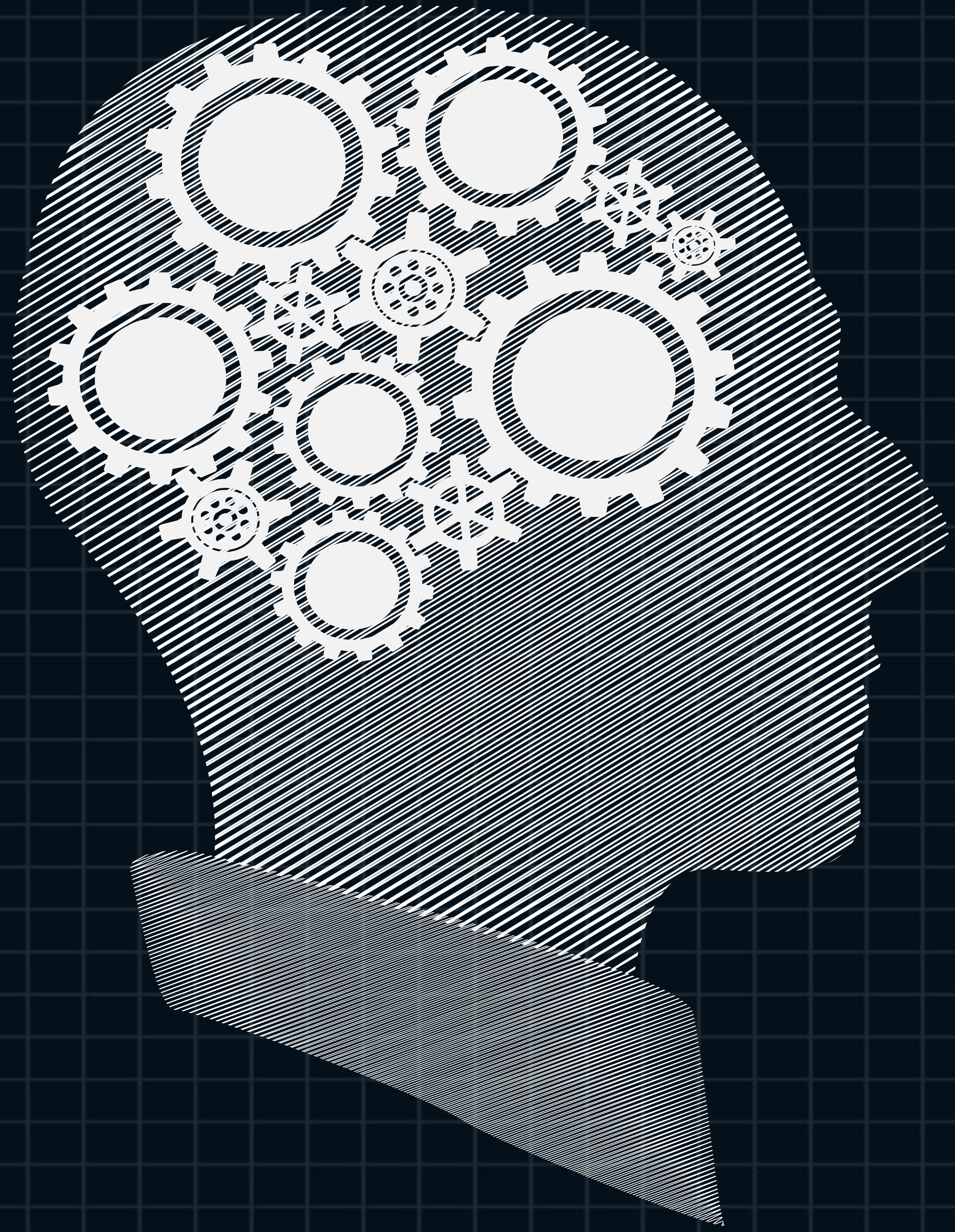


NumPy Master Class

Lecture.13 Dimensionality Manipulations

Lecture.13
Dimensionality Manipulations - Making New Dimensions

Using np.reshape

```python
import numpy as np

a = np.arange(9)


b = a.reshape((1, 9))
c = a.reshape((9, 1))

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

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

```
a: (9,)
 [0 1 2 3 4 5 6 7 8]
b: (1, 9)
 [[0 1 2 3 4 5 6 7 8]]
```

```
c: (9, 1)
 [[0]
 [1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]]
```

Lecture.13
Dimensionality Manipulations

- Making New Dimensions

Using np.reshape

```python
import numpy as np

a = np.arange(9)


b = a.reshape((1, -1))
c = a.reshape((-1, 1))


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

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

```
a: (9,)
 [0 1 2 3 4 5 6 7 8]

b: (1, 9)
 [[0 1 2 3 4 5 6 7 8]]
```

```
c: (9, 1)
 [[0]
  [1]
  [2]
  [3]
  [4]
  [5]
  [6]
  [7]
  [8]]
```

Lecture.13
Dimensionality Manipulations - Making New Dimensions

Using np.reshape

```python
import numpy as np

a = np.arange(9)

b = a.reshape((1, 1, 9))
c = a.reshape((1, 9, 1))
d = a.reshape((9, 1, 1))

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

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

```
a: (9,)
 [0 1 2 3 4 5 6 7 8]
```

```
b: (1, 1, 9)
 [[[0 1 2 3 4 5 6 7 8]]]


c: (1, 9, 1)
 [[[0]
  [1]
  [2]
  [3]
  [4]
  [5]
  [6]
  [7]
  [8]]]
```

```
d: (9, 1, 1)
 [[[0]]

 [[1]]

 [[2]]

 [[3]]

 [[4]]

 [[5]]

 [[6]]

 [[7]]

 [[8]]]
```

- Making New Dimensions

Using np.reshape

```
import numpy as np

a = np.random.normal(size=(100, 200))


b = a.reshape((1, 100, 200))
c = a.reshape((100, 200, 1))


print(b.shape)      (1, 100, 200)
print(c.shape)      (100, 200, 1)
```

- Making New Dimensions

Using np.reshape

```
a = (10, 20)
print(*a)                10 20
```

```
import numpy as np

a = np.random.normal(size=(100, 150))

print(a.shape)                  (100, 150)

print(*a.shape)                 100 150

print((1, *a.shape))            (1, 100, 150)

print((*a.shape, 1))            (100, 150, 1)
```

- Making New Dimensions

Using np.reshape

```
import numpy as np

a = np.random.normal(size=(100, 200))


b = a.reshape((1, *a.shape))
c = a.reshape((*a.shape, 1))


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

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

```
a.shape: (100, 200)

b.shape: (1, 100, 200)
c.shape: (100, 200, 1)
```

- Making New Dimensions

Using Slicing

```
import numpy as np

a = np.arange(9)

row_vec1 = a[np.newaxis, :]
row_vec2 = a[None, :]

col_vec1 = a[:, np.newaxis]
col_vec2 = a[:, None]

print(f"row_vec1.shape: {row_vec1.shape}")
print(f"row_vec2.shape: {row_vec2.shape}\n")

print(f"col_vec1.shape: {col_vec1.shape}")
print(f"col_vec2.shape: {col_vec2.shape}")
```

```
row_vec1.shape: (1, 9)
row_vec2.shape: (1, 9)

col_vec1.shape: (9, 1)
col_vec2.shape: (9, 1)
```

Lecture.13
Dimensionality Manipulations - Making New Dimensions

Using Slicing

```python
import numpy as np

a = np.arange(9)


b = a[np.newaxis, np.newaxis, :]
c = a[np.newaxis, :, np.newaxis]
d = a[:, np.newaxis, np.newaxis]


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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (9,)

b.shape: (1, 1, 9)
c.shape: (1, 9, 1)
d.shape: (9, 1, 1)
```

- Making New Dimensions

Using Slicing

```python
import numpy as np

a = np.random.normal(size=(100, 200))


b = a[np.newaxis, ...]
c = a[..., np.newaxis]

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

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

```
a.shape: (100, 200)

b.shape: (1, 100, 200)
c.shape: (100, 200, 1)
```

- Making New Dimensions

Using expand_dims API

```python
import numpy as np

a = np.arange(9)


b = np.expand_dims(a, axis=0)
c = np.expand_dims(a, axis=1)

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

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

```
a.shape: (9,)

b.shape: (1, 9)
c.shape: (9, 1)
```

- Making New Dimensions

Using expand_dims API

```
import numpy as np

a = np.arange(9)


b = np.expand_dims(a, axis=(0, 1))
c = np.expand_dims(a, axis=(0, 2))
d = np.expand_dims(a, axis=(1, 2))

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (9,)

b.shape: (1, 1, 9)
c.shape: (1, 9, 1)
d.shape: (9, 1, 1)
```

- Making New Dimensions

Using expand_dims API

```
import numpy as np

a = np.arange(9).reshape((3, 3))

b = np.expand_dims(a, axis=0)
c = np.expand_dims(a, axis=1)
d = np.expand_dims(a, axis=-1)
e = np.expand_dims(a, axis=(0, -1))

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
print(f"e.shape: {e.shape}")
```

```
a.shape: (3, 3)

b.shape: (1, 3, 3)
c.shape: (3, 1, 3)
d.shape: (3, 3, 1)
e.shape: (1, 3, 3, 1)
```

- Removing Dummy Dimensions

Using np.reshape

```python
import numpy as np

a = np.ones(shape=(1, 10))


b = a.reshape((10, ))
c = a.reshape((-1, ))
d = a.flatten()

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (1, 10)

b.shape: (10,)
c.shape: (10,)
d.shape: (10,)
```

Lecture.13
Dimensionality Manipulations

- Removing Dummy Dimensions

Using np.reshape

```python
import numpy as np

a = np.ones(shape=(10, 1))

b = a.reshape((10, ))
c = a.reshape((-1, ))
d = a.flatten()

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (10, 1)

b.shape: (10,)
c.shape: (10,)
d.shape: (10,)
```

- Removing Dummy Dimensions

Using np.reshape

```python
import numpy as np

a = np.ones(shape=(1, 3, 4))
b = np.ones(shape=(3, 4, 1))

c = a.reshape(*a.shape[1:])
d = b.reshape(*b.shape[:-1])

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

print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (1, 3, 4)
b.shape: (3, 4, 1)

c.shape: (3, 4)
d.shape: (3, 4)
```

Lecture.13
Dimensionality Manipulations

- Removing Dummy Dimensions

Using Slicing

```python
import numpy as np

a = np.arange(9).reshape((3, 3))

row, col = a[1, :], a[:, 1]

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

print(f"row.shape: {row.shape}")
print(f"col.shape: {col.shape}")
```

```
a.shape: (3, 3)

row.shape: (3,)
col.shape: (3,)
```

Lecture.13
Dimensionality Manipulations

- Removing Dummy Dimensions

Using Slicing

```
import numpy as np

a = np.arange(9).reshape((1, -1))
b = np.arange(9).reshape((-1, 1))

c = a[0, :]
d = b[:, 0]

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

print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (1, 9)
b.shape: (9, 1)

c.shape: (9,)
d.shape: (9,)
```

Lecture.13
Dimensionality Manipulations

- Removing Dummy Dimensions

Using Slicing

```python
import numpy as np

a = np.ones(shape=(1, 3, 4))
b = np.ones(shape=(3, 4, 1))

c = a[0, ...]
d = b[..., 0]

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

print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (1, 3, 4)
b.shape: (3, 4, 1)

c.shape: (3, 4)
d.shape: (3, 4)
```

Lecture.13
Dimensionality Manipulations

- Removing Dummy Dimensions

Using squeeze API

```
import numpy as np

a = np.ones(shape=(1, 3, 4))


c = np.squeeze(a)
d = a.squeeze()

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

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

```
a.shape: (1, 3, 4)

c.shape: (3, 4)
d.shape: (3, 4)
```

- Removing Dummy Dimensions

Using squeeze API

```
import numpy as np

a = np.ones(shape=(1, 1, 4, 1, 3, 1))

c = np.squeeze(a)
d = a.squeeze()

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

print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (1, 1, 4, 1, 3, 1)

c.shape: (4, 3)
d.shape: (4, 3)
```

- Changing Demensions

Using np.swapaxes API

```python
import numpy as np

a = np.random.normal(size=(3, 4, 5, 6))

b = np.swapaxes(a, 0, 1)
c = np.swapaxes(a, 0, 2)
d = np.swapaxes(a, 0, 3)

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (3, 4, 5, 6)

b.shape: (4, 3, 5, 6)
c.shape: (5, 4, 3, 6)
d.shape: (6, 4, 5, 3)
```

- Changing Demensions

Using np.swapaxes API

```
import numpy as np

a = np.random.normal(size=(3, 200, 100))

b = np.swapaxes(a, 0, -1)

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

print(f"b.shape: {b.shape}")
```

```
a.shape: (3, 200, 100)

b.shape: (100, 200, 3)
```

- Changing Demensions

Using np.moveaxis API

```python
import numpy as np

a = np.random.normal(size=(3, 4, 5, 6))


b = np.moveaxis(a, source=0, destination=1)
c = np.moveaxis(a, source=0, destination=2)
d = np.moveaxis(a, source=0, destination=-1)

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (3, 4, 5, 6)

b.shape: (4, 3, 5, 6)
c.shape: (4, 5, 3, 6)
d.shape: (4, 5, 6, 3)
```

- Changing Demensions

Using np.moveaxis API

```
import numpy as np

a = np.random.normal(size=(3, 4, 5, 6))


b = np.moveaxis(a, source=1, destination=0)
c = np.moveaxis(a, source=1, destination=2)
d = np.moveaxis(a, source=1, destination=-1)

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
```

```
a.shape: (3, 4, 5, 6)

b.shape: (4, 3, 5, 6)
c.shape: (3, 5, 4, 6)
d.shape: (3, 5, 6, 4)
```

- Changing Demensions

Using np.moveaxis API

```python
import numpy as np

a = np.random.normal(size=(3, 4, 5, 6))


b = np.moveaxis(a, source=0, destination=-1)
c = np.moveaxis(b, source=-2, destination=0)


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

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

```
a.shape: (3, 4, 5, 6)

b.shape: (4, 5, 6, 3)
c.shape: (6, 4, 5, 3)
```

- Changing Demensions

Using np.transpose API

```
import numpy as np

a = np.random.normal(size=(3, 4))


b = np.transpose(a)
c = a.T


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

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

```
a.shape: (3, 4)

b.shape: (4, 3)
c.shape: (4, 3)
```

- Changing Demensions

Using np.transpose API

```
import numpy as np

a = np.random.normal(size=(3, 4, 5, 6, 7))


b = np.transpose(a)
c = a.T


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

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

```
a.shape: (3, 4, 5, 6, 7)

b.shape: (7, 6, 5, 4, 3)
c.shape: (7, 6, 5, 4, 3)
```

- Changing Demensions

Using np.transpose API

```python
import numpy as np

a = np.random.normal(size=(3, 4, 5))


b = np.transpose(a, axes=(0, 1, 2))
c = np.transpose(a, axes=(1, 2, 0))
d = np.transpose(a, axes=(2, 0, 1))
e = np.transpose(a, axes=(2, 1, 0))

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

print(f"b.shape: {b.shape}")
print(f"c.shape: {c.shape}")
print(f"d.shape: {d.shape}")
print(f"e.shape: {e.shape}")
```

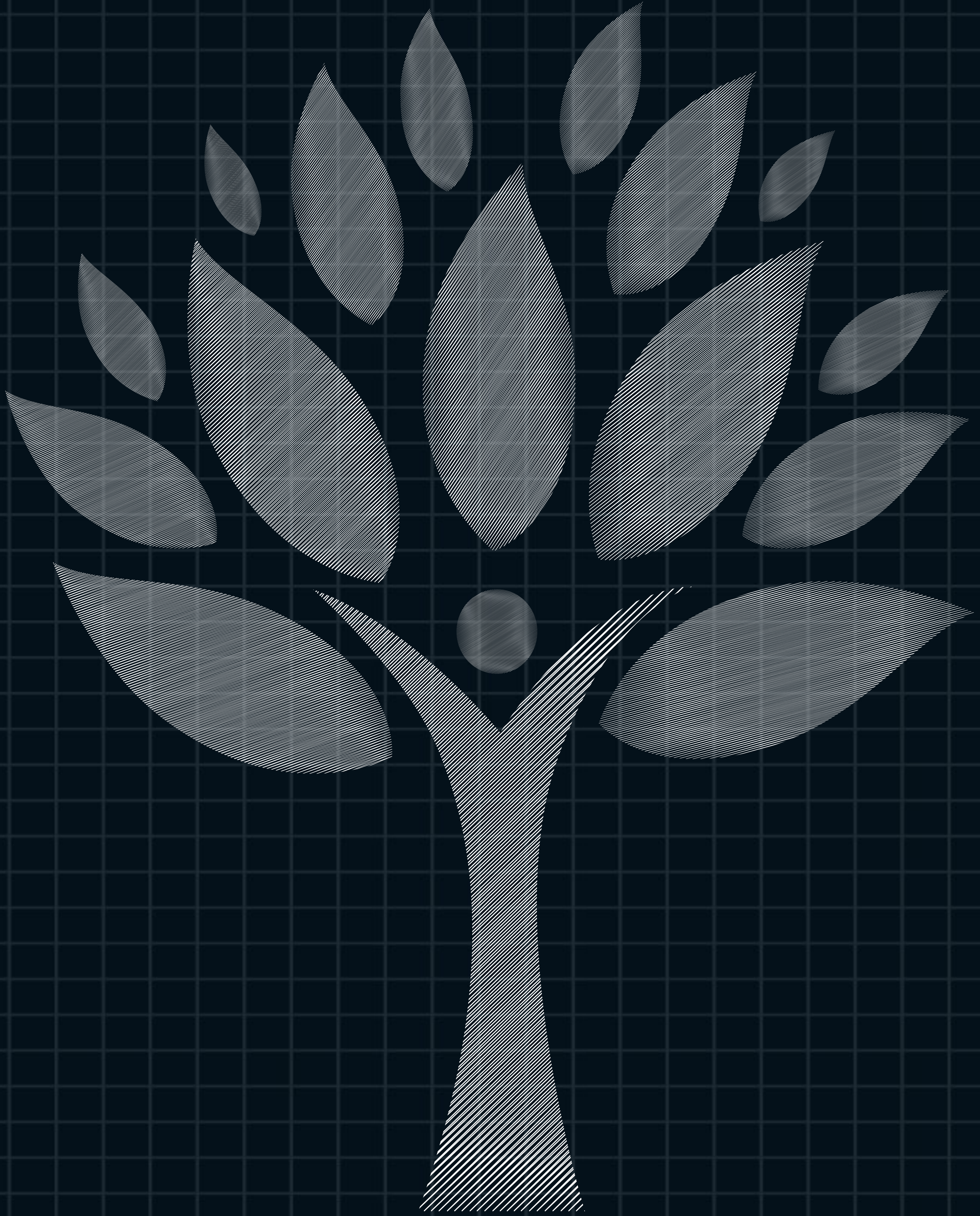
```
a.shape: (3, 4, 5)

b.shape: (3, 4, 5)
c.shape: (4, 5, 3)
d.shape: (5, 3, 4)
e.shape: (5, 4, 3)
```

- Changing Demensions

Using np.transpose API

```
import numpy as np

a = np.random.normal(size=(3, 4, 5, 6))

b = np.transpose(a, axes=tuple(range(a.ndim))[::-1])


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

print(f"b.shape: {b.shape}")
```

```
a.shape: (3, 4, 5, 6)

b.shape: (6, 5, 4, 3)
```



NumPy Master Class

Lecture.13 Dimensionality Manipulations