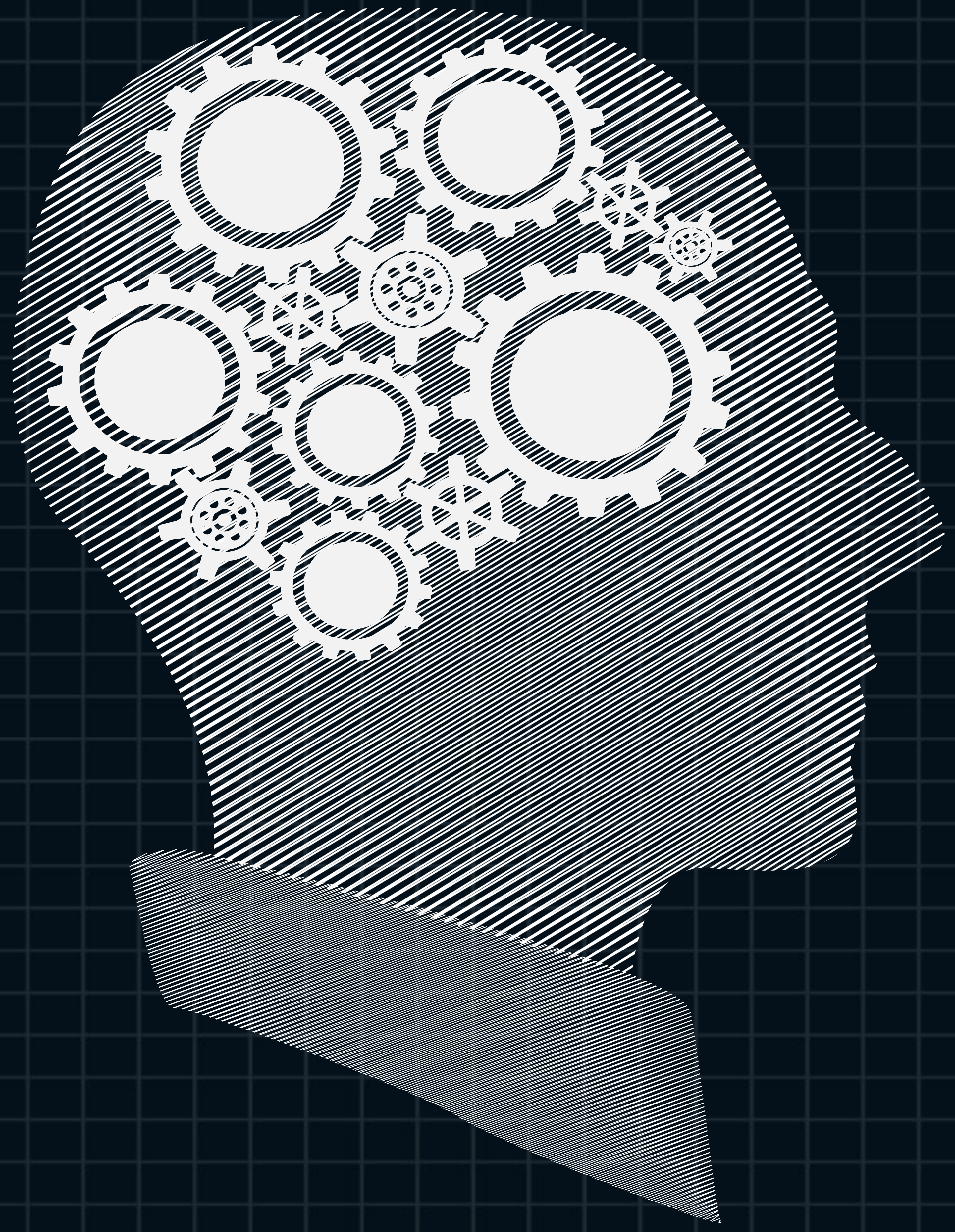


NumPy Master Class

Lecture.14
Merging ndarrays

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

np.hstack and np.vstack

```python
import numpy as np

a = np.random.randint(0, 10, (4, ))
b = np.random.randint(0, 10, (4, ))

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

```
a: (4,)
[4 2 1 0]
b: (4,)
[6 7 2 2]
```

```python
vstack = np.vstack([a, b])
hstack = np.hstack([a, b])

print(f"vstack: {vstack.shape}\n{vstack}")
print(f"hstack: {hstack.shape}\n{hstack}")
```

```
vstack: (2, 4)
[[4 2 1 0]
 [6 7 2 2]]
hstack: (8,)
[4 2 1 0 6 7 2 2]
```

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

np.hstack and np.vstack

```python
import numpy as np

a = np.random.randint(0, 10, (1, 3))
b = np.random.randint(0, 10, (1, 3))

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

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

```python
vstack = np.vstack((a, b))
hstack = np.hstack((a, b))

print(f"vstack: {vstack.shape}\n{vstack}")
print(f"hstack: {hstack.shape}\n{hstack}")
```

```
vstack: (2, 3)
[[0 8 7]
 [9 2 6]]
hstack: (1, 6)
[[0 8 7 9 2 6]]
```

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

np.hstack and np.vstack

```python
import numpy as np

a = np.random.randint(0, 10, (3, 1))
b = np.random.randint(0, 10, (3, 1))

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

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

```python
vstack = np.vstack((a, b))
hstack = np.hstack((a, b))

print(f"vstack: {vstack.shape}\n{vstack}")
print(f"hstack: {hstack.shape}\n{hstack}")
```

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

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

np.hstack and np.vstack

```python
import numpy as np

a = np.random.randint(0, 10, (3, 4))
b = np.random.randint(0, 10, (4,))

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

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

```python
vstack = np.vstack([a, b])

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

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

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

np.hstack and np.vstack

```python
import numpy as np

a = np.random.randint(0, 10, (3, 4))
b = np.random.randint(0, 10, (3,))

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

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

```python
hstack = np.hstack([a, b])

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

```
ValueError: all the input arrays must have same number of dimensions, but the array at index 0 has 2
dimension(s) and the array at index 1 has 1 dimension(s)
```

```python
hstack = np.hstack([a, b.reshape(-1, 1)])

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

```
hstack: (3, 5)
[[3 6 5 5 6]
 [9 5 9 0 0]
 [2 5 8 9 9]]
```

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

Making Toy Datasets

```python
import numpy as np

dataset = np.empty((0, 4))
print(f"initial shape: {dataset.shape}\n")

for iter in range(5):
  data_sample = np.random.uniform(0, 5, (1, 4))
  dataset = np.vstack((dataset, data_sample))
  print(f"iter/shape: {iter}/{dataset.shape}")
```

```
initial shape: (0, 4)


iter/shape: 0/(1, 4)
iter/shape: 1/(2, 4)
iter/shape: 2/(3, 4)
iter/shape: 3/(4, 4)
iter/shape: 4/(5, 4)
```

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

Making Toy Datasets

```python
import numpy as np

dataset = np.empty((4, 0))
print(f"initial shape: {dataset.shape}\n")

for iter in range(5):
  data_sample = np.random.uniform(0, 5, (4, 1))
  dataset = np.hstack((dataset, data_sample))
  print(f"iter/shape: {iter}/{dataset.shape}")
```

```
initial shape: (4, 0)

iter/shape: 0/(4, 1)
iter/shape: 1/(4, 2)
iter/shape: 2/(4, 3)
iter/shape: 3/(4, 4)
iter/shape: 4/(4, 5)
```

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

Making Toy Datasets(Efficient Way)

```python
import numpy as np

a = np.random.randint(0, 10, (1, 4))
b = np.random.randint(0, 10, (1, 4))
c = np.random.randint(0, 10, (1, 4))


arr_list = [a, b, c]
vstack = np.vstack(arr_list)


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

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

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

vstack: (3, 4)
[[4 1 0 3]
 [1 0 7 2]
 [3 2 8 5]]
```

Lecture.14
Merging ndarrays

- Merging Row/Column-wise

Making Toy Datasets(Efficient Way)

```
import numpy as np

dataset_tmp = list()
for iter in range(100):
  data_sample = np.random.uniform(0, 5, (1, 4))
  dataset_tmp.append(data_sample)

dataset = np.vstack(dataset_tmp)
print(f"final shape: {dataset.shape}")
```

```
final shape: (100, 4)
```

Lecture.14
Merging ndarrays

- Merging ndarrays using np.concatenate

np.concatenate

```python
import numpy as np

a = np.random.randint(0, 10, (3, ))
b = np.random.randint(0, 10, (4, ))

concat = np.concatenate([a, b])
concat0 = np.concatenate([a, b], axis=0)

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

print(f"concat.shape: {concat.shape}\n {concat}")
print(f"concat0.shape: {concat0.shape}\n {concat0}")
```

```
a: (3,)
 [1 9 9]
b: (4,)
 [2 6 9 4]

concat.shape: (7,)
 [1 9 9 2 6 9 4]
concat0.shape: (7,)
 [1 9 9 2 6 9 4]
```

Lecture.14
Merging ndarrays

- Merging ndarrays using np.concatenate

np.concatenate

```python
import numpy as np

a = np.random.randint(0, 10, (3, ))
b = np.random.randint(0, 10, (4, ))
c = np.random.randint(0, 10, (5, ))

concat = np.concatenate([a, b, c], axis=0)

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

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

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

concat.shape: (12,)
 [0 6 5 3 9 1 9 5 0 2 4 2]
```

- Merging ndarrays using np.concatenate

np.concatenate

```python
import numpy as np

a = np.random.randint(0, 10, (1, 3))
b = np.random.randint(0, 10, (1, 3))


axis0 = np.concatenate([a, b], axis=0)
axis1 = np.concatenate([a, b], axis=1)
axis_n1 = np.concatenate([a, b], axis=-1)


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

print(f"axis0: {axis0.shape}\n{axis0}")
print(f"axis1: {axis1.shape}\n{axis1}")
print(f"axis_n1: {axis_n1.shape}\n{axis_n1}")
```

```
a: (1, 3)
[[4 9 0]]
b: (1, 3)
[[8 1 0]]

axis0: (2, 3)
[[4 9 0]
 [8 1 0]]
axis1: (1, 6)
[[4 9 0 8 1 0]]
axis_n1: (1, 6)
[[4 9 0 8 1 0]]
```

- Merging ndarrays using np.concatenate

np.concatenate

```python
import numpy as np

a = np.random.randint(0, 10, (3, 4))
b = np.random.randint(0, 10, (3, 2))

concat = np.concatenate([a, b], axis=1)

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

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

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

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

Lecture.14
Merging ndarrays

- Merging ndarrays using np.concatenate

np.concatenate

```python
import numpy as np

a = np.random.randint(0, 10, (3, 4, 5))
```

```python
b = np.random.randint(0, 10, (10, 4, 5))
concat0 = np.concatenate([a, b], axis=0)
print(f"concat0: {concat0.shape}")
```

```
concat0: (13, 4, 5)
```

```python
b = np.random.randint(0, 10, (3, 10, 5))
concat1 = np.concatenate([a, b], axis=1)
print(f"concat1: {concat1.shape}")
```

```
concat1: (3, 14, 5)
```

```python
b = np.random.randint(0, 10, (3, 4, 10))
concat2 = np.concatenate([a, b], axis=2)
print(f"concat2: {concat2.shape}")
```

```
concat2: (3, 4, 15)
```

- Merging ndarrays using np.concatenate

Making Toy Datasets

```
import numpy as np

dataset_tmp = list()
for iter in range(100):
  data_sample = np.random.uniform(0, 5, (1, 4))
  dataset_tmp.append(data_sample)

concat = np.concatenate(dataset_tmp, axis=0)
print(f"concat: {concat.shape}")
```

```
concat: (100, 4)
```

```
dataset_tmp = list()
for iter in range(100):
  data_sample = np.random.uniform(0, 5, (4, 1))
  dataset_tmp.append(data_sample)

concat = np.concatenate(dataset_tmp, axis=1)
print(f"concat: {concat.shape}")
```

```
concat: (4, 100)
```

Lecture.14
Merging ndarrays

- Merging ndarrays Dimension-wise

np.dstack

```
import numpy as np

R = np.random.randint(0, 10, (100, 200))
G = np.random.randint(0, 10, size=R.shape)
B = np.random.randint(0, 10, size=R.shape)

image = np.dstack([R, G, B])

print(image.shape)   (100, 200, 3)
```

- Merging ndarrays Dimension-wise

np.dstack

```
import numpy as np

a = np.random.randint(0, 10, (100, 200, 3))
b = np.random.randint(0, 10, size=a.shape)
c = np.random.randint(0, 10, size=a.shape)

d = np.dstack([a, b, c])

print(d.shape)
```

```
(100, 200, 9)
```

Lecture.14
Merging ndarrays

- Merging ndarrays Dimension-wise

np.stack

```python
import numpy as np

a = np.random.randint(0, 10, (100, 200))
b = np.random.randint(0, 10, (100, 200))
c = np.random.randint(0, 10, (100, 200))

print("ndim==2:", np.stack([a, b, c]).shape)
```

```
ndim==2: (3, 100, 200)
```

```python
a = np.random.randint(0, 10, (100, 200, 300))
b = np.random.randint(0, 10, (100, 200, 300))
c = np.random.randint(0, 10, (100, 200, 300))

print("ndim==3:", np.stack([a, b, c]).shape)
```

```
ndim==3: (3, 100, 200, 300)
```

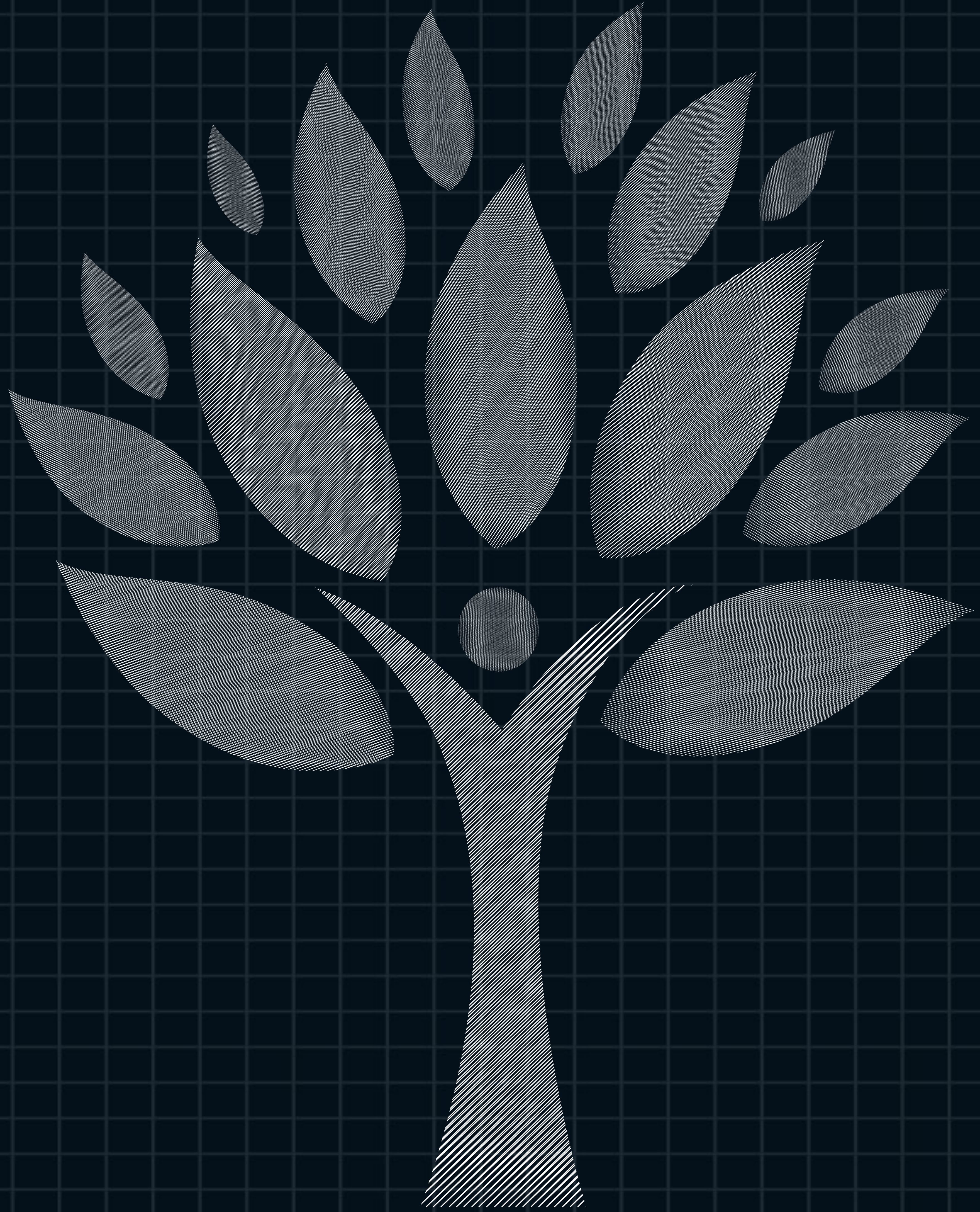
Lecture.14
Merging ndarrays

- Merging ndarrays Dimension-wise

np.stack

```
import numpy as np

a = np.random.randint(0, 10, (100, 200, 300))
b = np.random.randint(0, 10, (100, 200, 300))
c = np.random.randint(0, 10, (100, 200, 300))

print("axis=0:", np.stack([a, b, c], axis=0).shape)

print("axis=1:", np.stack([a, b, c], axis=1).shape)

print("axis=2:", np.stack([a, b, c], axis=2).shape)

print("axis=3:", np.stack([a, b, c], axis=3).shape)
```

```
axis=0: (3, 100, 200, 300)

axis=1: (100, 3, 200, 300)

axis=2: (100, 200, 3, 300)

axis=3: (100, 200, 300, 3)
```



NumPy Master Class

Lecture.14
Merging ndarrays