TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

LAB 7 QUESTION 3

By:

Sinjal Dahal (081/BEL/080)

DEPARTMENT OF COMPUTER ENGINEERING

LALITPUR, NEPAL

How to create an array

1. **Creating a 1D array**

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

Output:

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

2. **Creating a 2D array**

```
import numpy as np
b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(b)
```

Output:

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

How to access the elements

1. **Accessing specific elements**

```
import numpy as np
a = np.array([1, 2, 3, 4, 5, 6])
b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(a[0])
```

print(b[0, 0])

Output:

```
1
1
```

2. **Slicing**

import numpy as np

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

b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

print(a[0:2])

print(b[0:2, 0])

print(b[0:2, 0:2])

print(b[:, 2:])

Output:

```
[1 2]
[1 5]
[[1 2]
 [5 6]]
[[ 3  4]
 [ 7  8]
 [11 12]]
```

Basic array operations

1. **Element-wise operations**

import numpy as np

```
data = np.array([1, 2])
ones = np.ones(2, dtype=int)
print(data + ones)
print(data - ones)
print(data * ones)
print(data / ones)
```

Output:

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

2. **Multiplication and sum**

```
import numpy as np
print(data * 2)
print(data.sum())
```

Output:

```
[2 4]
3
```

3. **2D array operations**

```
import numpy as np
b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(b.sum(axis=0))
print(b.sum(axis=1))
```

Output:

```
[15 18 21 24]
[10 26 42]
```

Broadcasting

1. **Broadcasting example**

```
import numpy as np
data = np.array([1.0, 2.0])
print(data * 1.6)
```

Output:

```
[1.6 3.2]
```

More useful array operations

1. **Min, max, sum**

```
import numpy as np
data = np.array([1.0, 2.0])
print(data.max())
print(data.min())
print(data.sum())
```

Output:

```
2.0
1.0
3.0
```

2. **Standard deviation and product**

```
import numpy as np
```

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

Output:

```
1.707825127659933
720
```

Creating matrices

1. **Zeros, ones, and random arrays**

```
import numpy as np
print(np.zeros(2))
print(np.ones(2))
print(np.random.random(2))
```

Output:

```
[0. 0.]
[1. 1.]
[0.12345678 0.98765432]
```

Note: The random output varies per run.

2. **2D zeros and random arrays**

```
import numpy as np
print(np.zeros((2, 2)))
print(np.ones((2, 2)))
print(np.random.random((2, 2)))
```

Output:

```
[[0. 0.]
 [0. 0.]]
[[1. 1.]
 [1. 1.]]
[[0.45678901 0.23456789]
 [0.78912345 0.32165498]]
```

3. **Arange and reshape**

```
import numpy as np
print(np.arange(7))
print(np.arange(7).reshape(7, 1))
```

Output:

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

4. **Eye (identity matrix)**

```
import numpy as np
print(np.eye(3))
```

Output:

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

Unique items and counts

1. **Unique values and counts**

import numpy as np

array = np.array([11, 11, 12, 13, 14, 15, 16, 17, 12, 13, 11, 14, 18, 19, 20])

unique_values = np.unique(array)

print(unique_values)

unique_values, counts = np.unique(array, return_counts=True)

print(counts)

Output:

```
[11 12 13 14 15 16 17 18 19 20]
[3 2 2 2 1 1 1 1 1 1]
```

Transposing and reshaping

1. **Transpose**

import numpy as np

b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

print(b.T)

Output:

```
[[ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]
 [ 4  8 12]]
```

2. **Reshape**

```
import numpy as np
a = np.array([1, 2, 3, 4, 5, 6])
print(a.reshape(2, 3))
print(a.reshape(2, -1))
```

Output:

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

3. **Reverse array**

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
print(np.flip(arr))
```

Output:

```
[8 7 6 5 4 3 2 1]
```

4. **Reverse 2D array**

```
import numpy as np
```

```
array_2d = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(np.flip(array_2d))
print(np.flip(array_2d, axis=0))
print(np.flip(array_2d, axis=1))
```

Output:

```
[[12 11 10  9]
 [ 8  7  6  5]
 [ 4  3  2  1]]
[[ 9 10 11 12]
 [ 5  6  7  8]
 [ 1  2  3  4]]
[[ 4  3  2  1]
 [ 8  7  6  5]
 [12 11 10  9]]
```

How to inverse an array

1. **Inverse of a matrix**

```
import numpy as np
square_matrix = np.array([[1, 2], [3, 4]])
print(np.linalg.inv(square_matrix))
```

Output:

```
[[-2.   1. ]
 [ 1.5 -0.5]]
```

Flattening and raveling

1. **Flatten and ravel**

```
import numpy as np

array_2d = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

print(array_2d.flatten())

print(array_2d.ravel())
```

Output:

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
[ 1  2  3  4  5  6  7  8  9 10 11 12]
```

Save and load NumPy objects

1. **Saving and loading**

```
import numpy as np

array_2d = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

np.save('myarray.npy', array_2d)

loaded_array = np.load('myarray.npy')

print(loaded_array)
```

Output:

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```