TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

LAB 7 QUESTION 1

By:

Sinjal Dahal (081/BEL/080)

DEPARTMENT OF COMPUTER ENGINEERING

LALITPUR, NEPAL

arange_example

Documentation:

np.arange: Generates evenly spaced values within a given interval.

Parameters: start (number), stop (number), step (number, optional), dtype (data-type, optional).

Returns: array of evenly spaced values.

Example: np.arange(0, 10, 2) -> array([0, 2, 4, 6, 8])

Code:

```
import numpy as np
def arange_example():
    return np.arange(0, 10, 2)
print(arange_example())
```

Output:

```
[0 2 4 6 8]
```

linspace_example

Documentation:

np.linspace: Creates an array of evenly spaced numbers over a specified interval.

Parameters: start (number), stop (number), num (int, optional), dtype (data-type, optional).

Returns: array of evenly spaced numbers.

Example: np.linspace(0, 1, 5) -> array([0., 0.25, 0.5, 0.75, 1.])

Code:

```
import numpy as np
def linspace_example():
    return np.linspace(0, 1, 5)
print(linspace_example())
```

Output:

```
[0.   0.25 0.5  0.75 1.  ]
```

ones_example

Documentation:

np.ones: Creates a new array of the given shape and type, filled with ones.

Parameters: shape (int or tuple), dtype (data-type, optional).

Returns: array of ones.

Example: np.ones((2, 3)) -> array([[1., 1., 1.], [1., 1., 1.]])

Code:

```
import numpy as np
def ones_example():
    return np.ones((2, 3))
print(ones_example())
```

Output:

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

zeros_example

Documentation:

np.zeros: Creates a new array filled with zeros.

Parameters: shape (int or tuple), dtype (data-type, optional).

Returns: array of zeros.

Example: np.zeros((2, 3)) -> array([[0., 0., 0.], [0., 0., 0.]])

Code:

```
import numpy as np

def zeros_example():
    return np.zeros((2, 3))

print(zeros_example())
```

Output:

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

append_example

Documentation:

np.append: Appends values to the end of an array.

Parameters: arr (array-like), values (array-like), axis (int, optional).

Returns: array with appended values.

Example: np.append(np.array([1, 2]), [3, 4]) -> array([1, 2, 3, 4])

Code:

```
import numpy as np
def append_example():
    return np.append(np.array([1, 2]), [3, 4])
print(append_example())
```

Output:

```
[1 2 3 4]
```

concatenate_example

Documentation:

np.concatenate: Joins a sequence of arrays along an existing axis.

Parameters: arrays (sequence of arrays), axis (int, optional).

Returns: array of concatenated arrays.

Example: np.concatenate([np.array([1, 2]), np.array([3, 4])]) -> array([1, 2, 3, 4])

Code:

```
import numpy as np
def concatenate_example():
    return np.concatenate((np.array([1, 2]), np.array([3, 4])))
print(concatenate_example())
```

Output:

```
[1 2 3 4]
```

repeat_example

Documentation:

np.repeat: Duplicates elements in an array along a given axis.

Parameters: a (array-like), repeats (int or array of ints), axis (int, optional).

Returns: array with repeated elements.

Example: np.repeat(np.array([1, 2]), 2) -> array([1, 1, 2, 2])

Code:

```
import numpy as np
def repeat_example():
    return np.repeat(np.array([1, 2]), 2)
print(repeat_example())
```

Output:

```
[1 1 2 2]
```

reshape_example

Documentation:

np.reshape: Changes the shape of a NumPy array without altering its data.

Parameters: a (array-like), newshape (int or tuple).

Returns: reshaped array.

Example: np.reshape([1, 2, 3, 4], (2, 2)) -> array([[1, 2], [3, 4]])

Code:

```
import numpy as np

def reshape_example():

    return np.reshape(np.array([1, 2, 3, 4]), (2, 2))

print(reshape_example())
```

Output:

```
[[1 2]
 [3 4]]
```

resize_example

Documentation:

np.resize: Resizes an array and returns a new array with the specified size.

Parameters: a (array-like), new_shape (int or tuple).

Returns: array with specified size, repeating or truncating as needed.

Example: np.resize(np.array([1, 2, 3]), (5,)) -> array([1, 2, 3, 1, 2])

Code:

```
import numpy as np

def resize_example():

    return np.resize(np.array([1, 2, 3]), (5,))

print(resize_example())
```

Output:

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

split_example

Documentation:

np.split: Divides an array into separate sub-arrays along a specified axis.

Parameters: ary (array-like), indices_or_sections (int or array-like), axis (int, optional).

Returns: list of arrays.

Example: np.split(np.array([1, 2, 3, 4]), 2) -> [array([1, 2]), array([3, 4])]

Code:

```
import numpy as np
def split_example():
    return np.split(np.array([1, 2, 3, 4]), 2)
print(split_example())
```

Output:

```
[array([1, 2]), array([3, 4])]
```

transpose_example

Documentation:

np.transpose: Alters the dimensional arrangement of an array by reversing or swapping its axes.

Parameters: a (array-like), axes (tuple, optional).

Returns: transposed array.

Example: np.transpose(np.array([[1, 2], [3, 4]])) -> array([[1, 3], [2, 4]])

Code:

```
import numpy as np
def transpose_example():

    return np.transpose(np.array([[1, 2], [3, 4]]))
print(transpose_example())
```

Output:

```
[[1 3]
 [2 4]]
```

amax_example

Documentation:

np.amax: Returns the maximum of a given array or maximum along an axis.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.amax(np.array([1, 5, 3])) -> 5

Code:

```
import numpy as np
def amax_example():
    return np.amax(np.array([1, 5, 3]))
print(amax_example())
```

Output:

```
5
```

amin_example

Documentation:

np.amin: Returns the minimum of an array or minimum along an axis.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.amin(np.array([1, 5, 3])) -> 1

Code:

```
import numpy as np
def amin_example():
    return np.amin(np.array([1, 5, 3]))
print(amin_example())
```

Output:

```
1
```

argmax_example

Documentation:

np.argmax: Returns the indices of the maximum values along a specified axis.

Parameters: a (array-like), axis (int, optional).

Returns: array of indices.

Example: np.argmax(np.array([1, 5, 3])) -> 1

Code:

```
import numpy as np
def argmax_example():
    return np.argmax(np.array([1, 5, 3]))
print(argmax_example())
```

Output:

```
1
```

argmin_example

Documentation:

np.argmin: Finds the index of the minimum value in an array or along a specified axis.

Parameters: a (array-like), axis (int, optional).

Returns: array of indices.

Example: np.argmin(np.array([1, 5, 3])) -> 0

Code:

```
import numpy as np
def argmin_example():
    return np.argmin(np.array([1, 5, 3]))
print(argmin_example())
```

Output:

```
0
```

log_example

Documentation:

np.log: Returns an element-wise natural logarithm for an array.

Parameters: x (array-like).

Returns: array.

Example: np.log(np.array([1, np.e, np.e**2])) -> array([0., 1., 2.])

Code:

```
import numpy as np

def log_example():

    return np.log(np.array([1, np.e, np.e**2]))

print(log_example())
```

Output:

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

max_example

Documentation:

np.max: Returns the maximum value of an array or along a specified axis.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.max(np.array([1, 5, 3])) -> 5

Code:

```
import numpy as np
def max_example():
    return np.max(np.array([1, 5, 3]))
print(max_example())
```

Output:

```
5
```

mean_example

Documentation:

np.mean: Calculates the arithmetic mean of elements in an array along the specified axis.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.mean(np.array([1, 2, 3, 4])) -> 2.5

Code:

```
import numpy as np
def mean_example():
    return np.mean(np.array([1, 2, 3, 4]))
print(mean_example())
```

Output:

```
2.5
```

median_example

Documentation:

np.median: Returns the median of the elements in a given array.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.median(np.array([1, 2, 3, 4])) -> 2.5

Code:

```
import numpy as np
def median_example():
    return np.median(np.array([1, 2, 3, 4]))
print(median_example())
```

Output:

2.5

min_example

Documentation:

np.min: Finds the minimum value in an array or along a specified axis.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.min(np.array([1, 5, 3])) -> 1

Code:

```
import numpy as np
def min_example():
    return np.min(np.array([1, 5, 3]))
print(min_example())
```

Output:

```
1
```

nonzero_example

Documentation:

np.nonzero: Returns the indices of the non-zero values in a given array.

Parameters: a (array-like).

Returns: tuple of arrays.

Example: np.nonzero(np.array([0, 1, 0, 2])) -> (array([1, 3]),)

Code:

```
import numpy as np
def nonzero_example():
    return np.nonzero(np.array([0, 1, 0, 2]))
print(nonzero_example())
```

Output:

```
(array([1, 3]),)
```

percentile_example

Documentation:

np.percentile: Computes the q-th percentile of data along a specified axis.

Parameters: a (array-like), q (float or array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.percentile(np.array([1, 2, 3, 4]), 50) -> 2.5

Code:

```
import numpy as np
def percentile_example():
    return np.percentile(np.array([1, 2, 3, 4]), 50)
print(percentile_example())
```

Output:

```
2.5
```

sort_example

Documentation:

np.sort: Returns a sorted copy of an array in ascending order.

Parameters: a (array-like), axis (int, optional), kind (str, optional).

Returns: sorted array.

Example: np.sort(np.array([3, 1, 2])) -> array([1, 2, 3])

Code:

```
import numpy as np
def sort_example():
    return np.sort(np.array([3, 1, 2]))
print(sort_example())
```

Output:

```
[1 2 3]
```

sum_example

Documentation:

np.sum: Sums the elements of an array over a given axis.

Parameters: a (array-like), axis (int or tuple, optional).

Returns: array or scalar.

Example: np.sum(np.array([[1, 2], [3, 4]])) -> 10

Code:

```
import numpy as np
def sum_example():
    return np.sum(np.array([[1, 2], [3, 4]]))
print(sum_example())
```

Output:

```
10
```

where_example

Documentation:

np.where: Returns elements from arrays depending on a condition.

Parameters: condition (array-like), x (array-like, optional), y (array-like, optional).

Returns: array or tuple of arrays.

Example: np.where(np.array([1, 2, 3]) > 2, 1, 0) -> array([0, 0, 1])

Code:

```
import numpy as np
def where_example():
    return np.where(np.array([1, 2, 3]) > 2, 1, 0)
print(where_example())
```

Output:

```
[0 0 1]
```

det_example

Documentation:

np.linalg.det: Computes the determinant of a square matrix.

Parameters: a (array-like, square matrix).

Returns: float.

Example: np.linalg.det(np.array([[1, 2], [3, 4]])) -> -2.0

Code:

```
import numpy as np
def det_example():
    return np.linalg.det(np.array([[1, 2], [3, 4]]))
print(det_example())
```

Output:

```
-2.0000000000000004
```

dot_example

Documentation:

np.dot: Computes the dot product of two arrays.

Parameters: a (array-like), b (array-like).

Returns: array or scalar.

Example: np.dot(np.array([1, 2]), np.array([3, 4])) -> 11

Code:

```
import numpy as np
def dot_example():
    return np.dot(np.array([1, 2]), np.array([3, 4]))
print(dot_example())
```

Output:

```
11
```

inv_example

Documentation:

np.linalg.inv: Inverts a given square matrix.

Parameters: a (array-like, square matrix).

Returns: array.

Example: np.linalg.inv(np.array([[1, 2], [3, 4]])) -> array([[-2., 1.], [1.5, -0.5]])

Code:

```
import numpy as np

def inv_example():
    return np.linalg.inv(np.array([[1, 2], [3, 4]]))
print(inv_example())
```

Output:

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

trace_example

Documentation:

np.trace: Calculates the sum of the elements along the main diagonal of an array.

Parameters: a (array-like), offset (int, optional).

Returns: array or scalar.

Example: np.trace(np.array([[1, 2], [3, 4]])) -> 5

Code:

```
import numpy as np
def trace_example():
    return np.trace(np.array([[1, 2], [3, 4]]))
print(trace_example())
```

Output:

```
5
```