

Tutorial Week 2: NumPy for Structured Data (Solutions)

[Click to download the Jupyter Notebook files \(.ipynb\)](#)

0.1 Example: Element-wise addition of 2 NumPy arrays

Given are 2 NumPy arrays with the same shape, return a new array, in which every element is an element-wise sum of the 2 arrays.

```
[1]: import numpy as np

a1 = np.array([[1,2,3],
               [4,5,6]])

a2 = np.array([[10,11,12],
               [13,14,15]])

### Start your code here ###

b = a1 + a2
print(b)

### End your code here ###
```

```
[[11 13 15]
 [17 19 21]]
```

1 Questions

Hint:

- Write **your code** between two comment lines: `### Start/End your code here ###`.
- **Expected output** is shown at the end of each question (directly below the code cell).

1.1 Create a 2D NumPy array

- Create a 5x2 integer array from a range between 100 and 200 such that the difference between each element is 10.
- Print the array and its shape.

Hint:

- Use NumPy `arange()` and `reshape()`.

```
[2]: import numpy as np

### Start your code here ###
```

```
a = np.arange(100, 200, 10)
b = a.reshape(5,2)
print("The array is:")
print(b)
print("Its shape is: ")
print(b.shape)

### End your code here ###
```

The array is:

```
[[100 110]
 [120 130]
 [140 150]
 [160 170]
 [180 190]]
```

Its shape is:

```
(5, 2)
```

1.2 Reverse a 1D array

Write a function to reverse an array (the first element becomes the last).

```
[3]: import numpy as np

print("Original array:")
a = np.array([1, 2, 5, 10, 15, 86])
print(a)

print("Reverse array:")

### Start your code here ###

b = a[::-1]
print(b)

### End your code here ###
```

Original array:

```
[ 1  2  5 10 15 86]
```

Reverse array:

```
[86 15 10  5  2  1]
```

1.3 Multiply a 2D array by a scalar

Given a 2D array (matrix), return an array, which is equal to the original matrix multiplied by 2.

```
[4]: import numpy as np

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

```
### Start your code here ###  
  
# multiplying the numpy array a(matrix) by 2  
b = 2*a  
print(b)  
  
### End your code here ###
```

```
[[ 2  4  6]  
 [ 8 10 12]]
```

1.4 Horizontal stacking of NumPy arrays

- Stack 2 NumPy arrays horizontally i.e., 2 arrays having the same 1st dimension (number of rows in 2D arrays).
- Print the array and its shape.

```
[5]: import numpy as np  
  
a1 = np.array([[1,2,3],  
               [4,5,6]])  
  
a2 = np.array([[7,8,9],  
               [10,11,12]])  
  
### Start your code here ###  
  
b = np.hstack((a1, a2))  
print("The stacked array is: ")  
print(b)  
print("Its shape is: ")  
print(b.shape)  
  
### End your code here ###
```

The stacked array is:

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

Its shape is:

```
(2, 6)
```

1.5 Vertically stacking of NumPy arrays

- Stack 2 NumPy arrays vertically i.e., 2 arrays having the same last dimension (number of columns in 2D arrays).
- Print the array and its shape.

```
[6]: import numpy as np  
  
a1 = np.array([[1,2],  
               [3,4],  
               [5,6]])
```

```

a2 = np.array([[7,8],
               [9,10],
               [10,11]])

### Start your code here ###

b = np.vstack((a1, a2))
print("The stacked array is: ")
print(b)
print("Its shape is: ")
print(b.shape)

### End your code here ###

```

The stacked array is:

```

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

```

Its shape is:

```

(6, 2)

```

1.6 Convert the values of Celsius degrees into Fahrenheit degrees

Convert the values of Celsius degrees (C) into Fahrenheit degrees (F). Values are stored in a NumPy array and rounded to 2 decimal places.

$$F = 9 * \frac{C}{5} + 32$$

```

[7]: import numpy as np
C = np.array([-27.79, -11.12, 7.34, 1.16, 37.73, 0.70])
print("Values in Centigrade degrees:")
print(C)

print("Values in Fahrenheit degrees:")

### Start your code here ###

F = np.round((9*C/5 + 32), 2)
print(F)

### End your code here ###

```

Values in Centigrade degrees:

```

[-27.79 -11.12  7.34  1.16 37.73  0.7 ]

```

Values in Fahrenheit degrees:

```

[-18.02 11.98 45.21 34.09 99.91 33.26]

```

1.7 Print max from axis 0 and min from axis 1 from a 2D array

Hint:

- Use NumPy `amax()` and `amin()`.

```
[8]: import numpy as np

print("Original array:")
a = np.array([[34, 43, 73], [82, 22, 12], [53, 94, 66], [23, 45, 79]])
print(a)

### Start your code here ###

print("Max along axis 0:")
b = np.amax(a, axis=0)
print(b)

print("Min along axis 1:")
c = np.amin(a, axis=1)
print(c)

### End your code here ###
```

Original array:

```
[[34 43 73]
 [82 22 12]
 [53 94 66]
 [23 45 79]]
```

Max along axis 0:

```
[82 94 79]
```

Min along axis 1:

```
[34 12 53 23]
```

1.8 Select elements from a NumPy array which are divisible by 3

```
[9]: import numpy as np

print("Original array:")
a = np.array([5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29])
print(a)

print("Output array:")

### Start your code here ###

b = a[a%3==0]
print(b)

### End your code here ###
```

Original array:

```
[ 5  7  9 11 13 15 17 19 21 23 25 27 29]
```

Output array:
[9 15 21 27]

1.9 Select elements from a NumPy array which are greater than 5 and less than 20

```
[10]: import numpy as np

print("Original array:")
a = np.array([5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29])
print(a)

print("Output array:")

### Start your code here ###

b = a[(a > 5) & (a < 20)]
print(b)

### End your code here ###
```

Original array:

[5 7 9 11 13 15 17 19 21 23 25 27 29]

Output array:

[7 9 11 13 15 17 19]

1.10 Add, subtract, divide, and multiply 2D arrays

Hint:

- Use NumPy `add()`, `subtract()`, `divide()`, `multiply()`.

```
[11]: import numpy as np

a1 = np.array([[5, 10], [15, 20]])
a2 = np.array([[25, 30], [35, 40]])

### Start your code here ###

print ("Addition of two arrays:")
print (np.add(a1, a2))

print ("Subtraction of two arrays:")
print (np.subtract(a1, a2))

print ("Division of two arrays:")
print (np.divide(a1, a2))

print ("Multiplication of two arrays:")
print (np.multiply(a1, a2))

### End your code here ###
```

Addition of two arrays:

```
[[30 40]  
 [50 60]]
```

Subtraction of two arrays:

```
[[ -20  -20]  
 [ -20  -20]]
```

Division of two arrays:

```
[[0.2      0.33333333]  
 [0.42857143 0.5      ]]
```

Multiplication of two arrays:

```
[[125 300]  
 [525 800]]
```