**Numpy 2**

Mark as done

**NumPy Array Indexing**

**Access Array Elements**

Array indexing is the same as accessing an array element.

You can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

**Example**

Get the first element from the following array:

import numpy as np  
  
arr = np.array([1, 2, 3, 4])  
  
print(arr[0])

**Example**

Get the second element from the following array.

import numpy as np  
  
arr = np.array([1, 2, 3, 4])  
  
print(arr[1])

**Example**

Get third and fourth elements from the following array and add them.

import numpy as np  
  
arr = np.array([1, 2, 3, 4])  
  
print(arr[2] + arr[3])

**Access 2-D Arrays**

To access elements from 2-D arrays we can use comma separated integers representing the dimension and the index of the element.

Think of 2-D arrays like a table with rows and columns, where the dimension represents the row and the index represents the column.

**Example**

Access the element on the first row, second column:

import numpy as np  
  
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])  
  
print('2nd element on 1st row: ', arr[0, 1])

**Example**

Access the element on the 2nd row, 5th column:

import numpy as np  
  
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])  
  
print('5th element on 2nd row: ', arr[1, 4])

**Access 3-D Arrays**

To access elements from 3-D arrays we can use comma separated integers representing the dimensions and the index of the element.

**Example**

Access the third element of the second array of the first array:

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

**Negative Indexing**

Use negative indexing to access an array from the end.

**Example**

Print the last element from the 2nd dim:

import numpy as np  
  
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])  
  
print('Last element from 2nd dim: ', arr[1, -1])

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**NumPy Array Slicing**

**Slicing arrays**

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this:

[<em>start</em>:<em>**end**</em>]

.

We can also define the step, like this:

[<em>start</em>:<em>**end**</em>:<em>step</em>]

.

If we don't pass start its considered 0

If we don't pass end its considered length of array in that dimension

If we don't pass step its considered 1

**Example**

Slice elements from index 1 to index 5 from the following array:

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

**Note:** The result *includes* the start index, but *excludes* the end index.

**Example**

Slice elements from index 4 to the end of the array:

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

**Example**

Slice elements from the beginning to index 4 (not included):

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

**Negative Slicing**

Use the minus operator to refer to an index from the end:

**Example**

Slice from the index 3 from the end to index 1 from the end:

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

**STEP**

Use the

step

 value to determine the step of the slicing:

**Example**

Return every other element from index 1 to index 5:

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

**Example**

Return every other element from the entire array:

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

**Slicing 2-D Arrays**

**Example**

From the second element, slice elements from index 1 to index 4 (not included):

import numpy as np  
  
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])  
  
print(arr[1, 1:4])

**Note:** Remember that *second element* has index 1.

**Example**

From both elements, return index 2:

import numpy as np  
  
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])  
  
print(arr[0:2, 2])

**Example**

From both elements, slice index 1 to index 4 (not included), this will return a 2-D array:

import numpy as np  
  
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])  
  
print(arr[0:2, 1:4])

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### NumPy Data Types

## Data Types in Python

By default Python have these data types:

* strings

 - used to represent text data, the text is given under quote marks. e.g. "ABCD"

* integer

 - used to represent integer numbers. e.g. -1, -2, -3

* float

 - used to represent real numbers. e.g. 1.2, 42.42

* boolean

 - used to represent True or False.

* complex

 - used to represent complex numbers. e.g. 1.0 + 2.0j, 1.5 + 2.5j

## Checking the Data Type of an Array

The NumPy array object has a property called

dtype

 that returns the data type of the array:

### Example

Get the data type of an array object:

import numpy as np  
  
arr = np.array([1, 2, 3, 4])  
  
print(arr.dtype)

### Example

Get the data type of an array containing strings:

import numpy as np  
  
arr = np.array(['apple', 'banana', 'cherry'])  
  
print(arr.dtype)

# NumPy Sorting Arrays

## Sorting Arrays

Sorting means putting elements in an ordered sequence.

Ordered sequence is any sequence that has an order corresponding to elements, like numeric or alphabetical, ascending or descending.

The NumPy ndarray object has a function called

sort()

, that will sort a specified array.

### Example

Sort the array:

import numpy as np  
  
arr = np.array([3, 2, 0, 1])  
  
print(np.sort(arr))

**Note:** This method returns a copy of the array, leaving the original array unchanged.

You can also sort arrays of strings, or any other data type:

### Example

Sort the array alphabetically:

import numpy as np  
  
arr = np.array(['banana', 'cherry', 'apple'])  
  
print(np.sort(arr))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Statistical Functions in Python**

NumPy has quite a few useful statistical functions for finding minimum, maximum, percentile standard deviation and variance, etc from the given elements in the array. The functions are explained as follows −

Numpy is equipped with the robust statistical function as listed below

| **Function** | **Numpy** |
| --- | --- |
| Min | np.min() |
| Max | np.max() |
| Mean | np.mean() |
| Median | np.median() |
| Standard deviation | np.std() |

Consider the following Array:

**Example:**

import numpy as np

normal\_array = np.random.normal(5, 0.5, 10)

print(normal\_array)

**Output:**

[5.56171852 4.84233558 4.65392767 4.946659 4.85165567 5.61211317 4.46704244 5.22675736 4.49888936 4.68731125]

**Example of NumPy Statistical function**

### Min

print(np.min(normal\_array))

### Max

print(np.max(normal\_array))

### Mean

print(np.mean(normal\_array))

### Median

print(np.median(normal\_array))

### Sd

print(np.std(normal\_array))

**Output:**

4.467042435266913

5.612113171990201

4.934841002270593

4.846995625786663

0.3875019367395316