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: `[start:end]`.

We can also define the step, like this: `[start:end:step]`.

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])
```

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])
```

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])
```

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

Data Types in NumPy

NumPy has some extra data types, and refer to data types with one character, like i for integers, u for unsigned integers etc.

Below is a list of all data types in NumPy and the characters used to represent them.

- i - integer
- b - boolean
- u - unsigned integer
- f - float
- c - complex float
- m - timedelta
- M - datetime
- O- object
- S - string
- U - unicode string
- V - fixed chunk of memory for other type (void)

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)
```

Creating Arrays With a Defined Data Type

We use the **array()** function to create arrays, this function can take an optional argument: **dtype** that allows us to define the expected data type of the array elements:

Example

Create an array with data type string:

```
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='S')

print(arr)

print(arr.dtype)
```

Note- For i, u, f, S and U we can define size as well.

Example

Create an array with data type 4 bytes integer:

```
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='i4')

print(arr)

print(arr.dtype)
```

Converting Data Type on Existing Arrays

The best way to change the data type of an existing array, is to make a copy of the array with the astype() method.

The astype() function creates a copy of the array, and allows you to specify the data type as a parameter.

The data type can be specified using a string, like '**f**' for float, '**i**' for integer etc. or you can use the data type directly like **float** for float and **int** for integer.

Example

Change data type from float to integer by using 'i' as parameter value:

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype('i')

print(newarr)

print(newarr.dtype)
```

Example

Change data type from float to integer by using int as parameter value:

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype(int)

print(newarr)

print(newarr.dtype)
```

Example

Change data type from integer to boolean:

```
import numpy as np

arr = np.array([1, 0, 3])

newarr = arr.astype(bool)

print(newarr)

print(newarr.dtype)
```