Array From Existing Data - Tpoint Tech

numpy.asarray

This routine is used to create an array by using the existing data in the form of lists, or tuples. This routine is useful in the scenario where we need to convert a python sequence into the numpy array object.

The syntax to use the asarray() routine is given below.

1. numpy.asarray(sequence, dtype = None, order = None)

It accepts the following parameters.

1. **sequence:** It is the python sequence which is to be converted into the python array.
2. **dtype:** It is the data type of each item of the array.
3. **order:** It can be set to C or F. The default is C.

```
1. import numpy as np
2. l=[1,2,3,4,5,6,7]
3. a = np.asarray(l);
4. print(type(a))
5. print(a)
```

Output:

```
<class 'numpy.ndarray'>
[1 2 3 4 5 6 7]
```

```
1. import numpy as np
2. l=(1,2,3,4,5,6,7)
3. a = np.asarray(l);
4. print(type(a))
5. print(a)
```

Output:

```
<class 'numpy.ndarray'>
[1 2 3 4 5 6 7]
```

```
1. import numpy as np
2. l=[[1,2,3,4,5,6,7],[8,9]]
3. a = np.asarray(l);
4. print(type(a))
5. print(a)
```

Output:

```
<class 'numpy.ndarray'>
[list([1, 2, 3, 4, 5, 6, 7]) list([8, 9])]
```

numpy.frombuffer

This function is used to create an array by using the specified buffer. The syntax to use this buffer is given below.

```
numpy.frombuffer(buffer, dtype = float, count = -1, offset = 0)
```

It accepts the following parameters.

- **buffer:** It represents an object that exposes a buffer interface.
- **dtype:** It represents the data type of the returned data type array. The default value is 0.
- **count:** It represents the length of the returned ndarray. The default value is -1.
- **offset:** It represents the starting position to read from. The default value is 0.

```
import numpy as np
l = b'hello world'
print(type(l))
a = np.frombuffer(l, dtype = "S1")
print(a)
print(type(a))
```

Output:

```
<class 'bytes'>
[b'h' b'e' b'l' b'l' b'o' b' ' b'w' b'o' b'r' b'l' b'd']
<class 'numpy.ndarray'>
```

numpy.fromiter

This routine is used to create a ndarray by using an iterable object. It returns a one-dimensional ndarray object.

The syntax is given below.

```
numpy.fromiter(iterable, dtype, count = - 1)
```

It accepts the following parameters.

1. **Iterable:** It represents an iterable object.
2. **dtype:** It represents the data type of the resultant array items.
3. **count:** It represents the number of items to read from the buffer in the array.

```
import numpy as np
list = [0,2,4,6]
it = iter(list)
x = np.fromiter(it, dtype = float)
print(x)
print(type(x))
```

Output:

```
[0. 2. 4. 6.]
<class 'numpy.ndarray'>
```